circuitcellar.com

CYBERSECURITY
NOVEMBER 2025
ISSUE 424

circuit cellar

Inspiring the Evolution of Embedded Design

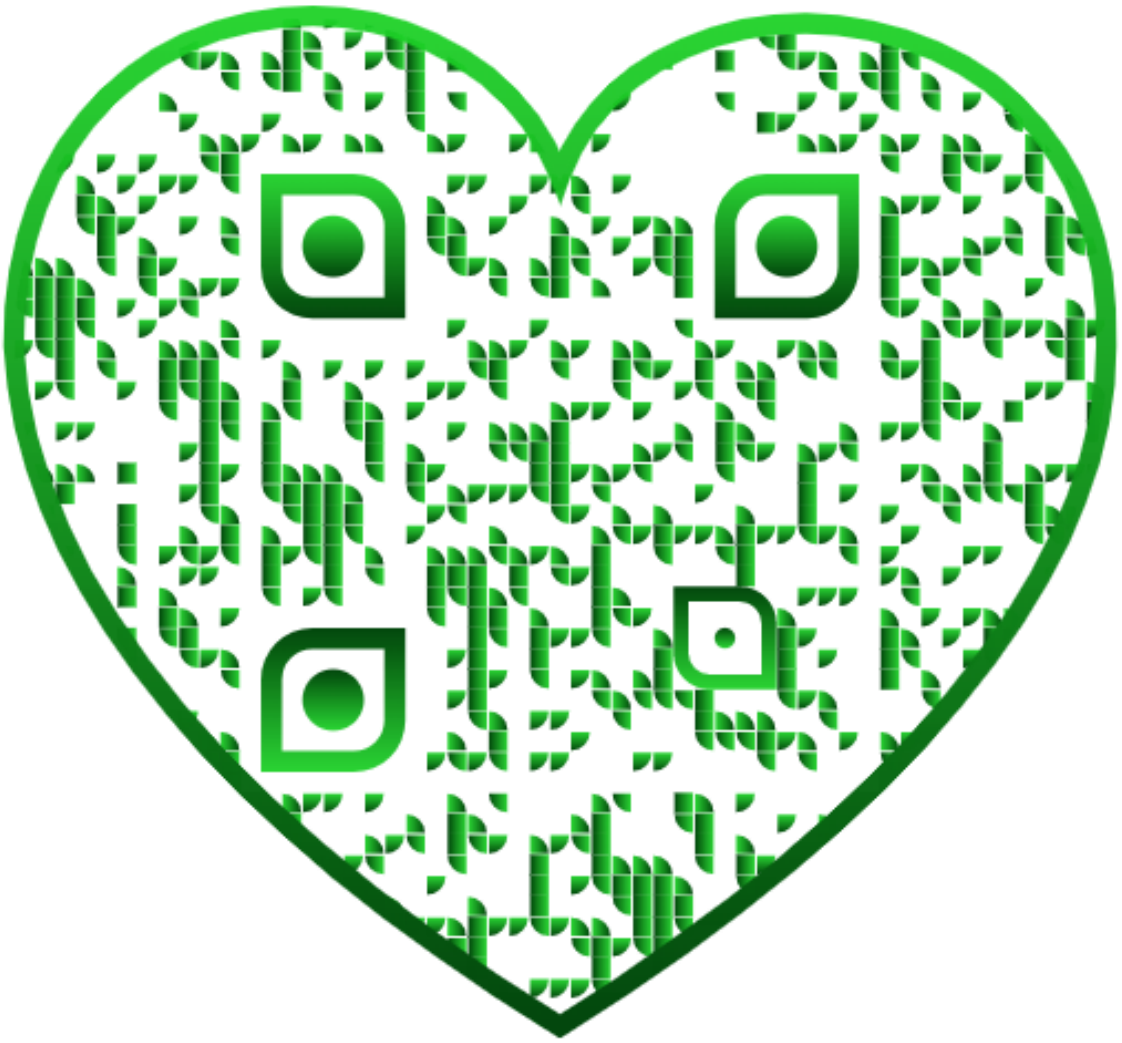
CYBERSECURITY

$9.00US $10.00CAN

11>

0 74470 75349 0

DC-to-DC Converters | Monitor MCU Resources on ESP32 Dashboard | Error and Uncertainty in Analog Electronics | Arduino-Controlled Gravitram Clock | Hand Motions for Environmental Controls | GPS-Disciplined Frequency Counter using the PSoC 5LP | Probe into Chip Silicon | The Future of Embedded Systems and AI



Imagine what you will

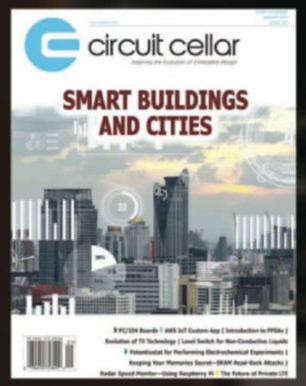
Discover

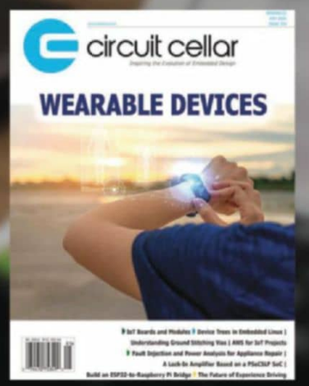
400+ ISSUES

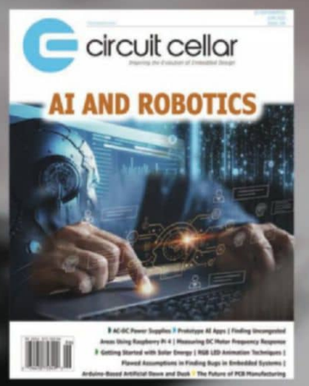
CC VAULT

Get the complete *Circuit Cellar* issue archive and article code stored on a stylish, durable and portable USB flash drive. You can easily keep your CC Vault archive up to date by purchasing subsequent issues from our webshop or by downloading issues with a *Circuit Cellar* Digital Subscription. Issues appear in searchable PDF format.

cc-webshop.com







Complete archive includes PDFs of all issues in print through date of purchase.

INPUT Voltage

It's a Language Thing — Does Your Favorite Tool Matter?

Do you have a favorite programming language? I don't necessarily mean the language you use most, but the language you reach for if you have several options—the language that just seems to directly translate from your brain to action. Most developers have one, and according to the Institute of Electrical and Electronic Engineers (IEEE), many of us are on the same page for 2025.

The 2025 IEEE Top Programming Languages Survey puts Python firmly at the top of the list, with Java coming in a solid second. C++ is number three, with a huge universe of languages following. Now, the IEEE was using research techniques that applied to enterprise IT programmers as well as embedded systems programmers, but all the evidence I've seen indicates that their conclusions apply to our industry as well as to the larger world of programming.

The real question that comes in response to the survey isn't "Is Python the best programming language?" but rather, "Will there be a need for programming languages in five years?" The growing number of software developers who build software via LLM AI prompts could make the idea of writing code directly as quaint as the concept of entering a program via control-panel toggle switches. The thing I'm wondering about is whether such a development would be a step forward or another move toward the abyss.

Back in My Day...Indulge me for a moment: Once upon a time, coding in assembly language (or even machine code) meant that you needed an intimate understanding of the computer you were programming in order to make something happen. High-level languages added a level of abstraction, trading reduced programming time for some efficiency and optimization in the program as it was running.

Low-code systems (e.g. Node-RED) and "vibe programming" with AI prompts add yet another abstraction layer — now, it's not necessary for a programmer to have any real understanding of how a device works. Now, all they really need to know is whether a microprocessor or SOC is capable of a particular action. The programming system takes their need and turns it into action.

Vastly more powerful processors supported by vastly more capable systems are what make these highly abstracted programming environments possible. They also make it, arguably, less important to wring the last jot of performance out of each clock cycle. We gain more productive software developers: What do we lose in return?

I'd like to know what you think. Is it important for software developers today to have a real understanding of what's going on inside a processor, or have modern programming systems made a higher level of knowledge sufficient? Are we finally close to the COBOL-era ideal of just telling the system what you want in simple human language? Or will there always be a need for programmers who have an understanding of the underlying hardware that's up close and personal? Please let me know your thoughts — I'll be sharing more of mine in upcoming issues.

In the meantime, I hope you'll be inspired by this month's articles on using embedded systems to improve communications for those with challenges, on getting the most out of an ESP32 system, and on the beginning steps of a journey into analog electronic design. As always, *Circuit Cellar* columnists show you some of the ins and outs of great projects. The truth is, there's a lot of great design and programming work going on, whether AI is involved or not.

Curtis Franklin
Editor-In-Chief
cfranklin@circuitcellar.com
LinkedIn: https://www.linkedin.com/in/curtisfranklin/
Facebook: https://www.facebook.com/curtis.f.franklin
Instagram: https://www.instagram.com/curt_franklin/
Mastodon: https://mastodon.sdf.org/@Kg4gwa
Bluesky: https://bsky.app/kg4gwa.bsky.social

Issue 424 November 2025 | ISSN 1528-0608

CIRCUIT CELLAR® (ISSN 1528-0608) is published monthly by:

KCK Media Corp.
PO Box 417, Chase City, VA 23924

Periodical rates paid at Chase City, VA, and additional offices. One-year (12 issues) subscription rate US and possessions $50, Canada $65, Foreign/ ROW $75. All subscription orders payable in US funds only via Visa, MasterCard, international postal money order, or check drawn on US bank.

SUBSCRIPTION MANAGEMENT

Online Account Management: circuitcellar.com/account
Renew | Change Address/E-mail | Check Status

CUSTOMER SERVICE

E-mail: customerservice@circuitcellar.com
Phone: 434.533.0246
Mail: Circuit Cellar, PO Box 417, Chase City, VA 23924

Postmaster: Send address changes to
Circuit Cellar, PO Box 417, Chase City, VA 23924

NEW SUBSCRIPTIONS

circuitcellar.com/subscription

ADVERTISING

Contact: Hugh Heinsohn
Phone: 757-525-3677
Fax: 888-980-1303
E-mail: hheinsohn@circuitcellar.com
Advertising rates and terms available on request.

NEW PRODUCTS

Contact: Kirsten Campbell
E-mail: product-editor@circuitcellar.com

HEAD OFFICE

KCK Media Corp.
PO Box 417
Chase City, VA 23924
Phone: 434-533-0246

COPYRIGHT NOTICE

Entire contents copyright © 2025 by KCK Media Corp. All rights reserved. Circuit Cellar is a registered trademark of KCK Media Corp. Reproduction of this publication in whole or in part without written consent from KCK Media Corp. is prohibited.

DISCLAIMER

KCK Media Corp. makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors printed in Circuit Cellar®. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects, KCK Media Corp. disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in Circuit Cellar®.

The information provided in Circuit Cellar® by KCK Media Corp. is for educational purposes. KCK Media Corp. makes no claims or warrants that readers have a right to build things based upon these ideas under patent or other relevant intellectual property law in their jurisdiction, or that readers have a right to construct or operate any of the devices described herein under the relevant patent or other intellectual property law of the reader's jurisdiction. The reader assumes any risk of infringement liability for constructing or operating such devices.

© KCK Media Corp. 2025 Printed in the United States

OUR NETWORK

circuit cellar · LinuxGizmos · audioxpress · VOICE COIL · LIS LOUDSPEAKER INDUSTRY SOURCEBOOK

SUPPORTING COMPANIES

Beningo Embedded Group	35
BusBoard Prototype Systems	9
EmbeddedTS	61, C4
Front Panel Express	25
MusicAI	19
RECOM	62
Saelig	31
SeaFire Micros, Inc.	C3
Siborg Systems, Inc.	61

NOT A SUPPORTING COMPANY YET?
Contact Hugh Heinsohn
hugh@circuitcellar.com,
Phone: 757-525-3677,
Fax: 888-980-1303
to reserve space in the
next issue of *Circuit Cellar.*

THE TEAM

FOUNDER	Steve Ciarcia
PUBLISHER	KC Prescott
CONTROLLER	Chuck Fellows
EDITOR-IN-CHIEF	Curtis Franklin
CONTRIBUTING EDITOR	Brian Millier
CONTENT MANAGER- EU	Jurij Mikeln
SENIOR ASSOCIATE EDITOR	Shannon Becker
TECHNICAL COPY EDITOR	Carol Bower
PROJECT EDITORS	Ken Davidson
	David Tweed
MARKETING MANAGER	Tori Zienka
ADVERTISING COORDINATOR	Diana Winsor
ADVERTISING SALES REP.	Hugh Heinsohn

COLUMNISTS
Stuart Ball (Start to Finish)
Jacob Beningo (Modern Embedded Insights)
Bob Japenga (Embedded in Thin Slices)
Brian Millier (Picking Up Mixed Signals)
Colin O'Flynn (Embedded Systems Essentials)

FEATURES

4 **An ESP32 Dashboard**
Monitoring How a Popular Microcontroller Uses Resources
By Pedro Bertoleti

10 **Designing Analog Electronics**
Part 1: Error and Uncertainty
By Andrew Levido

20 **Keeping Time with Rolling Balls**
An Arduino-Controlled Gravitram Clock
By Aubrey Kagan

26 **Empowering Disabled Persons with Hand Motions for Environmental Controls**
By Matthew Oppenheim

36 TECHNOLOGY FEATURE
CIA for SBCs (and More)
Securing Embedded Systems from Evolving Threats
By Curtis Franklin

42 DATASHEET
Making the Right Voltage
DC-to-DC Converters Bring the Right Voltage to Circuits
By Curtis Franklin

COLUMNS

46 Picking Up Mixed Signals
A GPS-Disciplined Frequency Counter Using the PSoC 5LP
By Brian Millier

54 Embedded System Essentials
Using a Desktop Laser Engraver to Probe into Chip Silicon
By Colin O'Flynn

63 TECH THE FUTURE
The Future of Embedded Systems and AI
AI in Embedded Systems
Cybersecurity Challenges and Opportunities in a Smarter World
By Pete Popov

58 : PRODUCT NEWS
62 : TEST YOUR EQ

@editor_cc @circuitcellar · circuitcellar · circuitcellar

An ESP32 Dashboard

Monitoring How a Popular Microcontroller Uses Resources

By
Pedro Bertoleti

Embedded devices are constrained by limited RAM restricted processing capacity, and finite, non-volatile storage. In this article, Pedro explains the importance of monitoring usage of an ESP32 microcontroller's resource allocation (including CPU/processing, RAM memory, heap usage, and stack usage) in real time, and how the FreeRTOS operating system optimizes ESP32 resource usage.

When designing embedded systems, one of the most critical challenges developers face is understanding and managing the microcontroller's hardware resources. Unlike high-end computing platforms, which typically feature abundant memory and processing power, embedded devices are known for being constrained by limited RAM, restricted processing capacity, and finite non-volatile storage. These limitations make it absolutely essential to continuously monitor how the system's resources are allocated and consumed in real-time. Failure to do so often leads to unpredictable malfunctions, performance degradation, or outright system crashes. Knowing how to measure the microcontroller CPU and memory resources, so as to extract every drop of computational power from them, is the key for successful embedded system development.

The ESP32 microcontroller family has gained massive popularity over the last 5 to 10 years due to its cost-effectiveness, integrated Wi-Fi and Bluetooth capabilities, good documentation and examples, and flexible development ecosystem. In many solutions that employ the ESP32, FreeRTOS serves as the underlying real-time operating system, providing task scheduling, inter-task communication, and resource management. This is because the ESP32 official software development framework (ESP-IDF) utilizes FreeRTOS out of the box. While FreeRTOS makes it easier to handle concurrency and modularity, it also introduces the need for resource monitoring, as multiple tasks run in parallel and compete for the same CPU, memory, and heap. Without real-time insights into how these resources are being utilized, developers risk pushing their applications into unstable territory. And this is the purpose of this article: to provide developers with tools to measure how intensively they are utilizing the ESP32's computational resources.

This article provides an in-depth technical guide on measuring CPU usage, RAM memory utilization, heap consumption, and stack allocation in an ESP32 running FreeRTOS. It not only introduces the APIs available for such monitoring, but also explains the theoretical background behind CPU and memory management in RTOS-based embedded software. By the end of this article, readers will understand why it is essential to track these parameters in real-time, and they will be able to utilize the necessary tools to integrate monitoring capabilities into their own ESP32-based solutions.

EMBEDDED SYSTEMS CPU USAGE

Let's start with one of the most important resources for embedded systems powered by a robust embedded software solution—the CPU usage of microcontrollers, more specifically, in those using real-time operating systems (RTOSes). The CPU is to a microcontroller what the engine is to a car, so monitoring how hard the CPU is being used in embedded systems is always a good idea.

In FreeRTOS, the embedded software is divided into small pieces of software called "tasks," where each task is responsible for doing one or a few functionalities of embedded software. In FreeRTOS, the CPU is shared among several tasks through a priority-based scheduler. Each task has a defined priority, and the scheduler ensures that the task with the highest priority among those ready to execute gains CPU access for a limited amount of time. The transition from one task execution to another is called "context change." As the context changes quickly (each task executes only for a few milliseconds), to us humans, it seems that all tasks are being executed in parallel, as shown in **Figure 1**.

The CPU usage in FreeRTOS, therefore, stands for the proportion of time the processor spends executing instructions of a given task, relative to the total available time. In percentage terms, summing across all tasks, CPU usage should not exceed 100%.

If the CPU load approaches its maximum capacity (100%) for extended periods, the system becomes overloaded and cannot accept new tasks or additional processing demands. In real-time systems, this results in severe consequences, including missed deadlines, jitter in periodic tasks, or degraded performance. For applications such as IoT devices, industrial monitoring, or wireless communication stacks, these problems directly reflect real-world failures, affecting IoT business as a whole.

Understanding CPU usage in FreeRTOS is essential, since it enables developers to guarantee that tasks with time constraints can execute reliably. By identifying which tasks consume the most CPU, optimizations can be prioritized in the right areas. Without such monitoring, developers are essentially guessing, which often leads to unstable deployments that perform well in short tests, but fail after prolonged use.

FIGURE 1
Tasks using CPU for a short amount of time. Source: FreeRTOS [1].

RAM DIVISION IN EMBEDDED SYSTEMS

The ESP32, as with all other microcontrollers, offers a limited amount of RAM that must be carefully managed across different uses in embedded software running RTOSes (including those using FreeRTOS as the operating system), memory can be categorized into three main areas: "static allocations," "stack memory," and "heap memory," as shown in **Figure 2**.

Static allocations are determined at compile time, when global or static variables are declared. The linker assigns them fixed memory positions, making them predictable and reliable, but also inflexible. Therefore, as a positive consequence, static allocation problems are shown at compile time, making it easier to detect and debug.

Stack memory is more dynamic. It refers to the amount of memory allocated for each task. Every task created in FreeRTOS has its own dedicated stack, which is allocated at runtime when the task is created. This stack is used to hold function call information, local variables, and return addresses. If the stack allocation for a task is too small, it risks overflowing, leading to corrupted memory and erratic behavior. These are only detectable in runtime tests, making it harder to detect and debug. If the stack allocation is too big, it implies bad usage of RAM memory (meaning waste of RAM memory in embedded

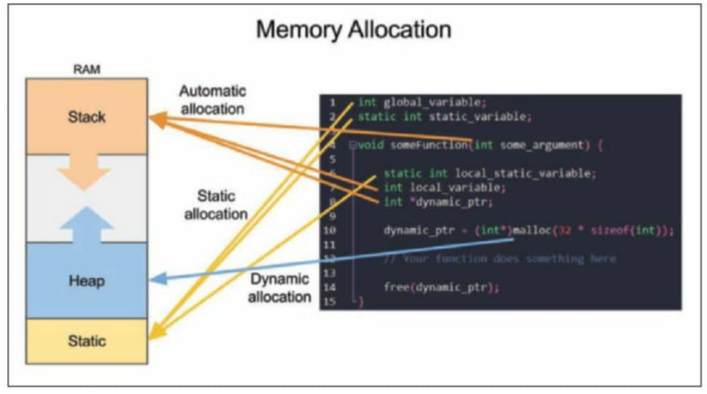


FIGURE 2
Memory allocation, highlighting Stack, Heap and Static Allocation. Source: DigiKey [2].

FEATURES

systems). To manage stack memory, the tasks' stack sizes must be wisely chosen and constantly monitored during the embedded software development phase.

The heap complements these allocations by serving as a pool of memory available for runtime allocation, typically used when modules or libraries require dynamic buffers or objects. While flexible, the heap introduces the risk of fragmentation, leaks, and unpredictable exhaustion, especially if not managed properly (for example, forgetting to free allocated memory after use). Heap memory can be a gift or a nightmare, depending on how its managed.

Thus, monitoring RAM memory usage and its divisions (static, stack, and heap) in real-time scope is not only about checking how much free memory remains globally, but also about tracking how each of these components is being consumed. One of the most practical metrics for stack memory in FreeRTOS is the "high-water mark," which will be discussed in detail later in this article.

IMPORTANCE OF MONITORING CPU USAGE IN REAL TIME

With the concepts of CPU and memory usage in embedded systems now clear, it's time to get deeper into the importance of monitoring each of them. Let's start with the importance of monitoring CPU usage in real-time.

Real-time CPU usage monitoring is critical because it ensures that tasks can be scheduled and executed within their expected time frames. If a system's CPU load is already near saturation (100%), even a minor increase in workload (a new task or even a new algorithm that uses nested loops) can destabilize the entire embedded software performance. Developers who actively measure CPU usage (and, of course, use this information to optimize CPU usage for developed tasks and algorithms) are helping to extract every drop of microcontroller CPU performance without compromising the entire embedded software.

Moreover, observing CPU usage helps prevent task starvation. In FreeRTOS, high-priority tasks will preempt lower-priority ones. If CPU utilization is consistently high, low-priority tasks may rarely execute. In practice, this can manifest as missed updates, unresponsive user interfaces, delayed communications, and malfunctions from user perspective. Monitoring CPU usage during the development phase enables the early detection of such problems, allowing developers to redistribute the workload or redesign scheduling strategies.

Types of CPU Usage to Monitor: Two types of CPU usage must be monitored. First, CPU Usage per Task. The APIs of RTOSes (and microcontrollers) allow measurement of what percentage of CPU each task is currently using. This allows developers to identify which tasks are using too much CPU, investigate the cause, and optimize the task to use less CPU.

Second, Overall CPU Usage. The APIs of RTOSes (and microcontrollers) allow developers to measure the percentage of CPU the embedded software is using. There can be two sub-metrics related to overall CPU usage—the sum of all tasks CPU usage, and its complement, the "IDLE" time (that is, the time the CPU spends doing nothing, waiting for the next task to execute). The higher the IDLE time, the lower the overall CPU usage of the entire embedded software being executed.

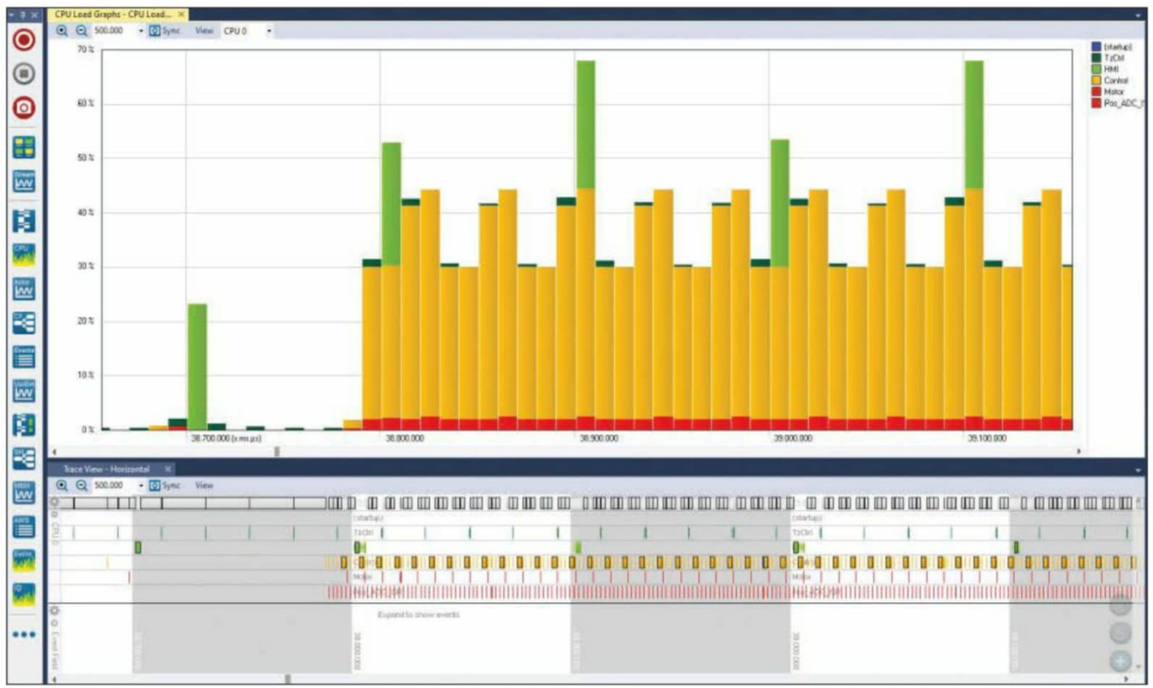


FIGURE 3
Overall CPU usage and CPU usage per task in Traceanalyzer tool. Source: FreeRTOS [4].

Figure 3 shows the overall CPU usage (higher left bar value for each time instant) and the CPU usage per task (represented by left bar amplitude for each color for each time instant, having one color per task) using the tool Traceanalyzer by Percepio [3].

REAL-TIME RAM USAGE

RAM consumption is closely tied to system reliability, and, therefore, is critical. Unlike CPU overload, which usually results in performance degradation, RAM usage issues often cause catastrophic crashes in embedded software. Monitoring RAM in real time allows developers to verify whether memory allocation decisions made during development hold true under real-world conditions.

This is particularly useful for long-running systems, since gradual increases in RAM usage over days or weeks of intense tests and usage reveal hidden problems such as memory leaks and memory fragmentation.

In FreeRTOS scope (and in this article), the importance of RAM usage monitoring will be split into two topics: the importance of monitoring heap use in real time, and the importance of monitoring task stack usage with high-water mark.

REAL TIME HEAP USAGE

Heap monitoring is indispensable, because heap allocation is "dynamic," meaning it occurs at runtime, and its consequences are not predictable at compile time. If heap usage grows steadily without stabilizing, it is almost always a sign of a memory leak. Similarly, even if total heap memory seems sufficient, memory fragmentation (caused by badly dimensioned heap memory allocations) may prevent large allocations from succeeding, and in some cases, may lead to embedded software failures and unpredictable behavior.

Despite the dynamic nature and unpredictability of heap memory allocation, tools in the APIs of microcontroller SDKs and RTOSes allow developers to determine exactly how heap memory is being utilized. For

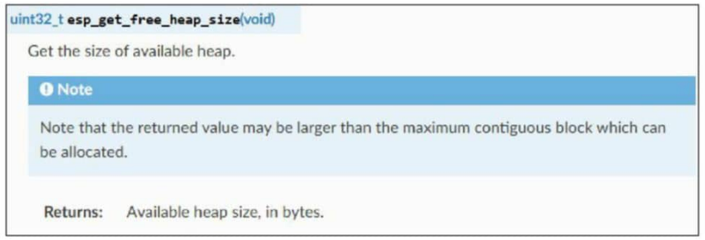

uint32_t esp_get_free_heap_size(void)
Get the size of available heap.
Note
Note that the returned value may be larger than the maximum contiguous block which can be allocated.
Returns: Available heap size, in bytes.

FIGURE 4
`esp_get_free_heap_size` shows the heap currently available API. Source: Espressif [5].

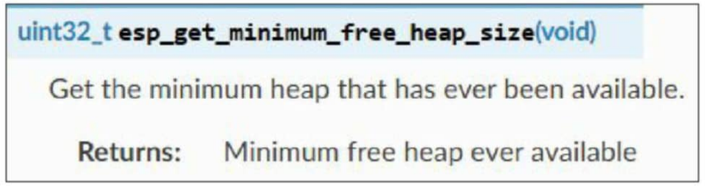

uint32_t esp_get_minimum_free_heap_size(void)
Get the minimum heap that has ever been available.
Returns: Minimum free heap ever available

FIGURE 5
`esp_get_minimum_free_heap_size` tells the lowest heap possible API. Source: Espressif [6].

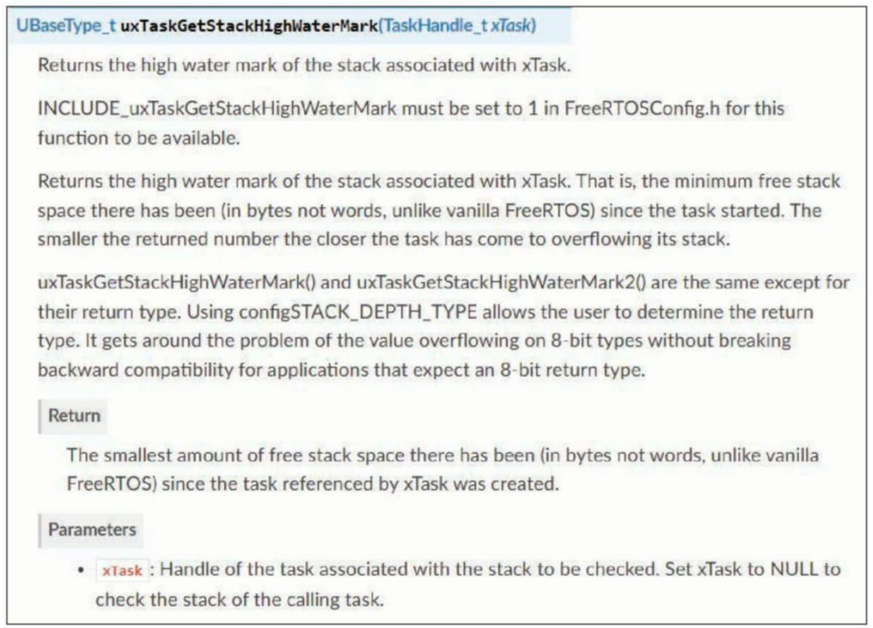

UBaseType_t uxTaskGetStackHighWaterMark(TaskHandle_t xTask)
Returns the high water mark of the stack associated with xTask.
INCLUDE_uxTaskGetStackHighWaterMark must be set to 1 in FreeRTOSConfig.h for this function to be available.
Returns the high water mark of the stack associated with xTask. That is, the minimum free stack space there has been (in bytes not words, unlike vanilla FreeRTOS) since the task started. The smaller the returned number the closer the task has come to overflowing its stack.
uxTaskGetStackHighWaterMark() and uxTaskGetStackHighWaterMark2() are the same except for their return type. Using configSTACK_DEPTH_TYPE allows the user to determine the return type. It gets around the problem of the value overflowing on 8-bit types without breaking backward compatibility for applications that expect an 8-bit return type.
Return
The smallest amount of free stack space there has been (in bytes not words, unlike vanilla FreeRTOS) since the task referenced by xTask was created.
Parameters
xTask : Handle of the task associated with the stack to be checked. Set xTask to NULL to check the stack of the calling task.

FIGURE 6
`xTaskGetStackHighWaterMark` is the most stack storage used by the routine. API. Source: Espressif [7].

ESP32, heap memory issues can be detected at runtime by using ESP-IDF out-of-the-box APIs such as `esp_get_free_heap_size()` and `esp_get_minimum_free_heap_size()`. These APIs are documented in **Figures 4 and 5**.

The first function, (`esp_get_free_heap_size`), reports the current amount of free heap available at any moment. The second one, (`esp_get_minimum_free_heap_size`), provides the lowest free heap memory value observed since system boot. It's important to use both APIs during intense and long-term tests, to get a reliable picture of how heap memory is being used.

Together, these two metrics in the ESP32 give both an instantaneous view and a historical minimum of heap memory allocation, allowing developers to know whether the system has ever approached critical conditions. Such insight is gold in terms of ensuring long-term reliability in production deployments.

HIGH-WATER MARK FOR TASK STACK USAGE

Beyond heap memory monitoring, it's crucial to monitor how tasks are using their stack memory, to determine if each task stack is sub-dimensioned or over-dimensioned.

To allow this kind of monitoring, the FreeRTOS provides a very powerful tool, the "high-water mark." In the FreeRTOS context, this term represents the minimum amount of stack space that has remained unused since a task started running. In other words, it indicates how close a task has come to completely filling its allocated stack.

The `uxTaskGetStackHighWaterMark()` ESP-IDF API is used for this purpose. It returns the smallest number of words (not bytes) that have remained unused in the stack since the task began its execution. By analyzing this value—especially after intense and long-term tests—developers can determine whether the stack allocation for a task is appropriate. If the high-water-mark value is very low, it means the task monitored has nearly overflowed its stack at some point, suggesting that the stack size should be increased to prevent stack smash under intense usage. If the high-water-mark value is consistently high, it indicates that the stack is over-provisioned and could be reduced safely, freeing RAM for other uses. To get a better idea on how this API works, see **Figure 6**.

FIGURE 7 vTaskGetRunTimeStats is a debugging tool for CPU monitoring at the runtime scope. API documentation. Source: FreeRTOS [8].

vTaskGetRunTimeStats

task.h

```
1  void vTaskGetRunTimeStats( char *pcWriteBuffer );
```

Copy

See the Run Time Stats page for a full description of this feature.

`configGENERATE_RUN_TIME_STATS`, `configUSE_STATS_FORMATTING_FUNCTIONS` and `configSUPPORT_DYNAMIC_ALLOCATION` must all be defined as 1 for this function to be available. The application must also then provide definitions for `portCONFIGURE_TIMER_FOR_RUN_TIME_STATS()` and `portGET_RUN_TIME_COUNTER_VALUE` to configure a peripheral timer/counter and return the timer's current count value respectively. The counter should be at least 10 times the frequency of the tick count.

NOTE: This function will disable interrupts for its duration. It is not intended for normal application runtime use but as a debug aid.

`vTaskGetRunTimeStats()` calls uxTaskGetSystemState(), then formats the raw data generated by `uxTaskGetSystemState()` into a human readable (ASCII) table that shows the amount of time each task has spent in the Running state (how much CPU time each task has consumed). The data is provided as both an absolute and a percentage value. The resolution of the absolute value is dependent on the frequency of the run time stats clock provided by the application.

`vTaskGetRunTimeStats()` is a utility function provided for convenience only. It is not considered part of the kernel. See vTaskList() for a utility function that generates information on the state of each task.

Parameters:

- *pcWriteBuffer*

  A buffer into which the execution times will be written, in ASCII form. This buffer is assumed to be large enough to contain the generated report. Approximately 40 bytes per task should be sufficient.

Additional materials from the author are available at: www.circuitcellar.com/article-materials

References [1] to [8] as marked in the article can be found there.

FREERTOS APIS MONITOR CPU USE

For CPU monitoring at a runtime scope, developers can use the FreeRTOS API vTaskGetRunTimeStats(), which is documentd in **Figure 7**.

This API reports the CPU usage of embedded software, showing the execution time of each task, and allowing detailed analysis of processing distribution. It also displays the IDLE metric, allowing developers to measure exactly what percentage of time the CPU spends waiting for the next task to execute.

After intense and long-term tests getting and analyzing the information available from `vTaskGetRunTimeStats` API, developers will see exactly how hard the CPU is being used and, most important, which tasks must be optimized in terms of CPU usage.

A BUILDING FUNDAMENTAL

Monitoring CPU, RAM, heap, and stack usage in real time is a fundamental aspect of building reliable and robust embedded systems with ESP32 and FreeRTOS, ensuring that the ESP32 resources are well and sufficient for the referred solution.

The CPU usage monitoring at runtime ensures that tasks meet their deadlines and that the system can scale without being overloaded, thereby preserving its functionalities and user experience. The RAM monitoring prevents unexpected crashes, while heap monitoring helps detect leaks and fragmentation issues at runtime. Stack monitoring at runtime through the high-water mark offers fine-grained visibility into task-level memory usage, preventing overflows and enabling optimal allocation of scarce resources.

By using specific APIs, such as:

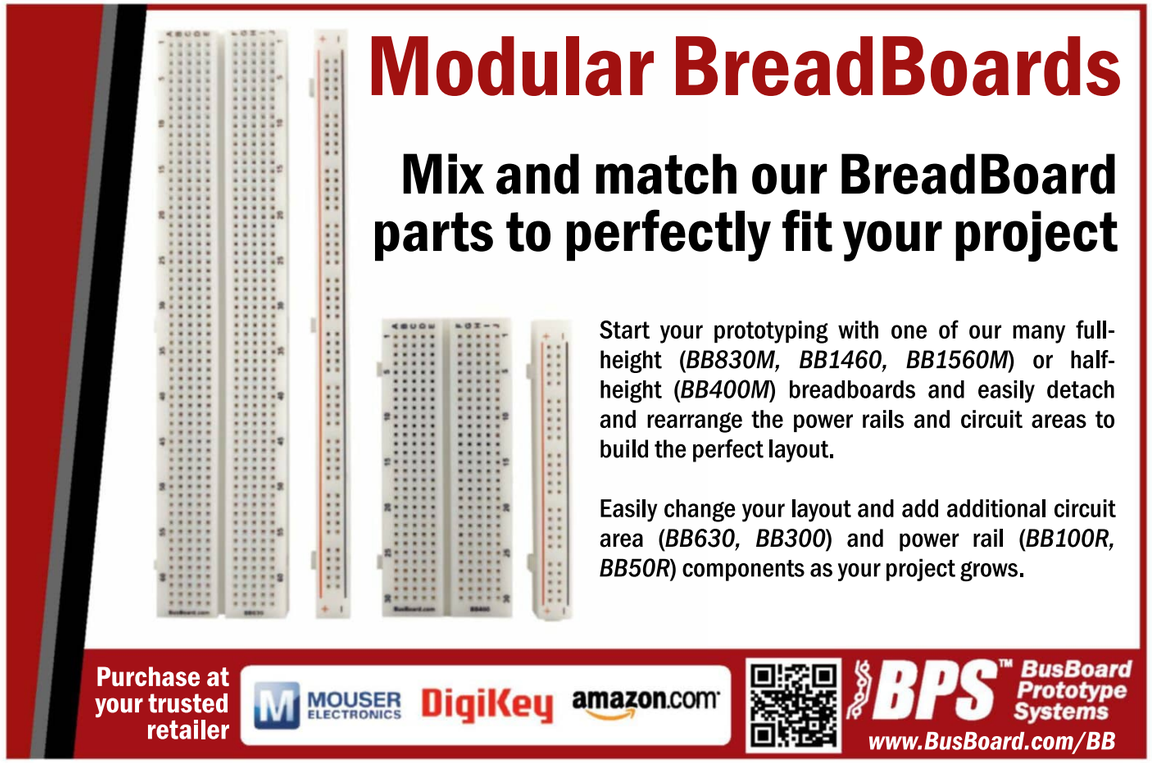
```
esp_get_free_heap_size()
esp_get_minimum_free_heap_size()
vTaskGetRunTimeStats()
uxTaskGetStackHighWaterMark()
```

developers gain the ability to analyze and optimize their system's performance throughout the development lifecycle. Incorporating these practices not only prevents runtime issues but also extends the longevity and reliability of deployed systems, which is particularly vital for embedded solutions expected to operate continuously and unattended in the field.

Uniting all these analyses and tools, robust and well-dimensioned embedded software can be developed for ESP32.

ABOUT THE AUTHOR

Pedro Bertoleti is a Brazilian embedded systems engineer and publisher on a Brazilian embedded systems website called Embarcados – www.embarcados.com.br. He has strong experience in development with PIC, ESP32, ARM Cortex M0 and M3 and RAK3172 microcontrollers. Bertoleti also has strong knowledge and experience in IoT projects and Embedded Linux Embedded Systems development.


Modular BreadBoards
Mix and match our BreadBoard parts to perfectly fit your project
Start your prototyping with one of our many full-height (BB830M, BB1460, BB1560M) or half-height (BB400M) breadboards and easily detach and rearrange the power rails and circuit areas to build the perfect layout.
Easily change your layout and add additional circuit area (BB630, BB300) and power rail (BB100R, BB50R) components as your project grows.
Purchase at your trusted retailer
MOUSER ELECTRONICS
DigiKey
amazon.com
BPS™ BusBoard Prototype Systems
www.BusBoard.com/BB

Designing Analog Electronics

Part 1: Error and Uncertainty

By
Andrew Levido

Designing analog electronics hardware is less orderly and predictable than digital electronics design. In this article series, Andrew describes the tools and techniques for producing high quality, reliable analog circuits. In Part 1, he looks at the key factor that differentiate analog and digital electronics—managing errors and uncertainty.

In the last 10 years or so, we have seen a real resurgence of interest in designing and building electronics hardware, driven by the accessibility of platforms such as Arduino and Raspberry Pi. Many of those people have come to hardware design from a software or firmware background, and have acquired the necessary skills by getting their hands dirty and building stuff.

One of the most challenging parts of hardware design (at least in my opinion) is analog design. This is an area of electronics which, while relatively straightforward to get started with, is really quite difficult to master. Unlike the order and predictability of digital electronics, analog electronics can be messy, noisy and uncertain. I firmly believe that there is no magic to good analog design. Armed with a little knowledge and a handful of tools and techniques, anyone can produce high-quality, reliable and reproducible analog circuits.

I hope to demonstrate some of these tools and techniques in this series of articles. They are aimed at readers who are comfortable with the basics of electronics, but want to strengthen their knowledge of analog circuit design. It is not meant to be a rigorous exposition of analog design theory, but rather to offer some tools, tips, and techniques that might help you avoid some of the traps and produce effective and functional analog circuits. I am going to keep things very practical, using a minimum of mathematics and lots of real-world examples.

A DESIGN EXAMPLE

To that end, we are going to begin with a simple, real-world example that comes up frequently. Let's assume we have a DC voltage, say in the range of 0-12V, that we want to measure with a microcontroller. A suitable block diagram is shown in **Figure 1**. The input voltage will be processed in the analog domain, then digitized by an analog-to-digital converter (ADC) within a microcontroller. The resulting ADC code is converted to a digital representation of the measured value that could be displayed or otherwise used by the application running on the microcontroller.

We will focus on the analog signal conditioning part, but we won't be able to do this without reference to the ADC and the digital signal conditioning, as you will see. I'm going to assume that there is a single 3.3V power supply available to power both the micro and the analog circuitry. Because the input is coming from an external source, we will design the circuit to withstand voltages up to ±100V indefinitely, without damage.

The microprocessor I have in mind has a 12-bit ADC, which means a theoretical precision of 1 bit in 2^{12}, or 0.025%. I am not so naïve as to think we can get this sort of

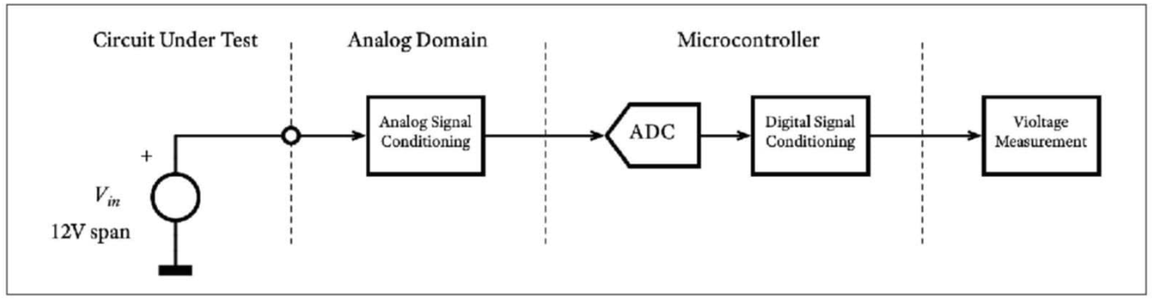


FIGURE 1
The simple task of digitizing a DC voltage as shown here is more complex than you might expect, if we are concerned about achieving a reasonable degree of precision.

precision overall (or even out of the ADC), but let's use this as a guide to shoot for and see what kind of resolution might be possible. This might feel like a fairly trivial design exercise, but I assure you, there is a lot to take into account if we want to get anywhere close to this level of precision.

ATTENUATOR

The first thing we have to do is to attenuate the input signal, so that its span (range of minimum voltage to maximum voltage) is reduced from 12V to around 3V to suit the ADC. Since we are dealing with DC signals, we can use a simple resistive voltage divider like the one shown in **Figure 2** to attenuate our signal by a factor of about 4.

I am being deliberately vague about the precise attenuation, because it does not matter much, as long as the span of the attenuator's output matches that of the ADC, so we make maximum use of the ADC's resolution. We will have to scale the digital code produced by the ADC to convert it to units of volts, so we can set this scaling factor to match the precise attenuation we end up using.

Changing the level (or amplitude) of a signal through attenuation or gain is one of the fundamental building blocks of analog circuit design. We define the gain of an analog stage as the ratio between its output amplitude and its input amplitude. If the gain is greater than 1, the circuit is an amplifier; and if it is less than 1, it is an attenuator. Attenuation is just the inverse of gain, so our attenuation target of 4 is the same as a gain of ¼.

In the divider of Figure 2, the output voltage is related to the input voltage by the familiar expression shown on the right. This gives us the ratio of resistor values that will provide the gain that we need, but not the values themselves. We will choose R1, so our divider has a relatively high impedance. In this way, we don't load down the source too much, and we limit the energy we will have to deal with under overvoltage conditions.

I chose R1 to be 2MΩ, so for a gain of 0.25 we would need R2 to be 667kΩ. The closest standard resistor value to this is 680kΩ, giving us a gain of 0.254—which should be fine. If we were to use this value in our digital signal conditioning block, we would get the correct readings, as long as the resistors R1 and R2 measure exactly 2MΩ and 680KΩ, respectively.

UNCERTAINTY AND ERROR

Unfortunately, there inevitably is some uncertainty in the resistance value, due to variations in the manufacturing process and environmental conditions. In fact, pretty much any component you care to name will have some level of error or uncertainty. The management of error and uncertainty is probably the biggest single factor that differentiates analog electronics from digital electronics. A large part of the process of analog design is the understanding, quantification and management of these errors. You will see some of this as we progress with our design.

Error or uncertainty can be expressed in relative terms, such as percent (parts per hundred) or parts per million (ppm), or in absolute terms such as Volts, Ohms, or Amperes. For example, if the 2MΩ resistor in our divider has a 1% tolerance, its absolute value could vary by ±20kΩ from the nominal

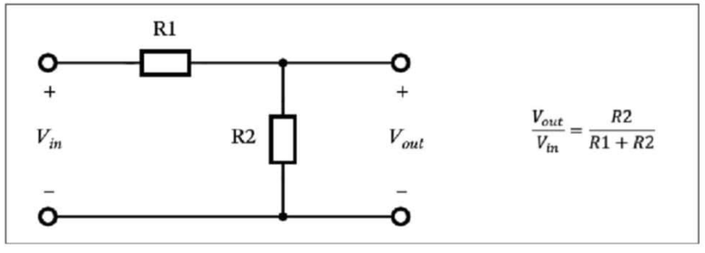


FIGURE 2
This simple restive divider attenuates the 12V input signal to one with a span compatible with the ADC. Resistor tolerance and temperature drift are potential sources of error we will have to manage.

value. The relative error is 1% and the absolute error is ±20kΩ. It is important to be able to switch between absolute and relative errors, because of the way errors accumulate.

When we add two quantities with errors, the error in the result is the sum of the absolute errors. In the case of our resistor divider, for example, the denominator of the gain expression is R1 + R2. The total resistance value will be 2.68MΩ (2MΩ + 680kΩ) and the error will be the sum of their absolute errors (±20kΩ + ±6.8kΩ = 26.8kΩ). Notice that the relative error here is still 1%.

The gain calculation requires us to divide the R2 by this sum. When we multiply or divide two quantities with errors, we add the relative errors, so the worst case divider gain error with 1% resistors will be ±2%. Adding relative errors when multiplying or dividing uncertain quantities is actually an approximation that assumes that both relative errors are fairly small (about 5% or less). If the relative errors are larger, this calculation will underestimate the error. The mathematical reasoning behind these useful rules of thumb is shown in **Figure 3**, for those interested.

The rule for multiplying and dividing values with errors also assumes the two variables are independent. However, in the case of our voltage divider, the numerator and the denominator share the R2 value. In practice this means that the worst case error of 2% will only occur if R1 is sufficiently larger than R2, such that R1 dominates the denominator.

If R1 and R2 are equal in value, the worst case error will be 1%. If R2 is much larger than R1, the error reduces further. Because our resistor values are not too far apart (they are in a roughly 3:1 ratio) the worst case error turns out to be just under 1.5%.

This is not even close to the precision we would like to achieve. Switching to 0.1% tolerance resistors helps, but the divider error will still be ±0.15%. We could use even higher tolerance resistors, but this gets expensive quickly. Fortunately, we have already had a glimpse of another solution. If we can set the firmware scaling factor to accommodate the actual voltage divider value, we could also tweak it for each device, to compensate for the tolerance error.

This is an example of "calibration," which is widely used in analog circuits to trim out fixed uncertainties. In the old days, we might have used a potentiometer for this, but today it is much more likely that calibration coefficients are applied in firmware. In our example there would be a calibration process where a precisely known voltage would be applied to the input, and the coefficients necessary to produce the correct output calculated and stored in non-volatile memory.

Adding two quantities with uncertainty

$$(z \pm \Delta z) = (x \pm \Delta x) + (y \pm \Delta y) = (x + y) \pm (\Delta x + \Delta y)$$

$$\therefore z = x + y \text{ and } \Delta z = \Delta x + \Delta y$$

Multiplying two quantities with uncertainty

$$(z \pm \Delta z) = (x \pm \Delta x)(y \pm \Delta y) = xy \pm (y\Delta x + x\Delta y + \Delta x\Delta y)$$

If the relative errors are small, the $\Delta x\Delta y$ term can be ignored

$$\therefore z = xy \text{ and } \Delta z \approx x\Delta y + y\Delta x$$

Dividing the error term by z gives the relative error

$$\frac{\Delta z}{z} \approx \frac{\Delta x}{x} + \frac{\Delta y}{y}$$

FIGURE 3
A really useful rule of thumb is: 1) add absolute errors when adding or subtracting two or more quantities with errors; and 2) add relative errors when multiplying or dividing such quantities. You can see here how these rules are derived.

TEMPERATURE COEFFICIENTS (TEMPCO)

There is, however, another potential problem that is not so easily overcome. Resistors also change value with temperature. This is a lesson I learned the hard way, as a very junior engineer. An equally junior colleague and I spent one long, hot summer afternoon carefully calibrating a batch of current-measuring meters for diesel-electric locomotives that we had previously assembled. Each was effectively an arc-shaped LED bar graph indicating the current level. No microcontrollers were involved, so calibration required lots of careful back-and-forth tweaking of potentiometers to get the zero and full-scale displays just right.

We left the meters powered on overnight, intending to pack them up for the customer the following morning. However, when we arrived in the cool early morning of the next day they were all showing exactly two lit LEDs despite there being no input signal. We had apparently built and calibrated a batch of very expensive thermometers! A senior engineer soon determined that some carbon film resistors had been mistakenly substituted for low-tempco metal film ones in critical locations. Thankfully, he was too polite to apportion blame. My colleague and I spent the remainder of that day changing all the resistors and repeating the whole tedious calibration process. Lesson learned.

Modern resistors are much better than old-school carbon film ones, but your typical 1% tolerance, thin-film resistor still has a tempco of ±100ppm/°C. This means the resistance can change 0.01% per degree Celsius. A resistor's nominal value is usually specified at 25°C, so if used in a device designed to operate over a temperature range of 0 to 50°C, these resistors will change value by up to ±0.25%. This error is in addition to the tolerance error.

This kind of error cannot be calibrated out as easily as just described, so it is often the case that an analog designer will pay more attention to the resistor's temperature coefficient than its tolerance. Resistors with tempcos of ±25ppm are not much more expensive, and may be worth considering for critical locations. Even lower tempcos are available at a price.

Before you despair of ever getting respectable precision out of a circuit like this, consider that in many cases (including our example), the absolute values of the two resistances are not as important as their ratio. Two resistors with the same temperature characteristic will maintain the same resistance ratio over a wide temperature range, provided they are both at the same temperature. In fact you can buy pairs or arrays of resistors built on a common substrate for just this purpose. They come in a fairly limited range of ratios, but have remarkable performance for the price.

For example I frequently use the Vishay ACASA range of thin-film resistor arrays containing two or four matched resistors in a small SMT package. They have an absolute tolerance of 0.1%, but a relative tolerance (matching) of 0.05%. They are available with absolute tempcos down to ±10ppm and relative tempcos down to ±5ppm. They cost just a couple of dollars each in small quantities.

Whether or not you need this kind of precision depends on your application. The important thing is to be aware of the potential sources of error and to be able to quantify them. Only then can you decide what you have to deal with and what you can ignore.

In our design we will use 2MΩ and 680kΩ resistors from the Susumu RGT series, which have a tolerance of 0.1% and a tempco of 10ppm. These are pretty good specs and for the price (about $0.16 each in small quantities). The worst case error due to resistor tolerance is therefore ±0.15%, which we can improve with calibration. I did not specify an operating temperature range for our example circuit, but assuming it is a laboratory device, 25±15°C should be sufficient. The worst case temperature error will add an additional ±0.015%, which we can live with, since it is lower than the ADC quantisation error.

SAMPLE AND HOLD

Unfortunately, we cannot just connect the output of the divider to the ADC's input. Because we have built a fairly high-impedance divider, it is likely we will need a buffer between the two. The data for the microcontroller I plan to use for this application (STM32L03)[1] provides a diagram of the equivalent circuit of an ADC input (**Figure 4**).

This shows that the ADC input is protected by two diodes, and has a leakage current of ±50nA. The effective source impedance of our divider (R_S) is around 500kΩ (2MΩ||680kΩ), so this leakage current would produce a voltage error of ±25mV (±0.8%), if the divider were directly connected. That's bad, but there is more.

The ADC is preceded by a multiplexor and a sample-and-hold circuit. During the sampling period the appropriate input is selected and the sample/hold switch is closed, so that the voltage on the sample capacitor (C_{SH}) charges to the source voltage. The switch is opened during the conversion period, to keep the converted voltage constant even if the input changes. The sampling switch and multiplexor all contribute some series resistance, which is lumped together and shown in the diagram as R_{SH}.

To sample the source voltage V_S accurately, the sum of R_S and R_{SH} must be low enough that the sampling capacitor charges fully during the time that the switch is closed. The source resistance must therefore be kept below some maximum level dependent

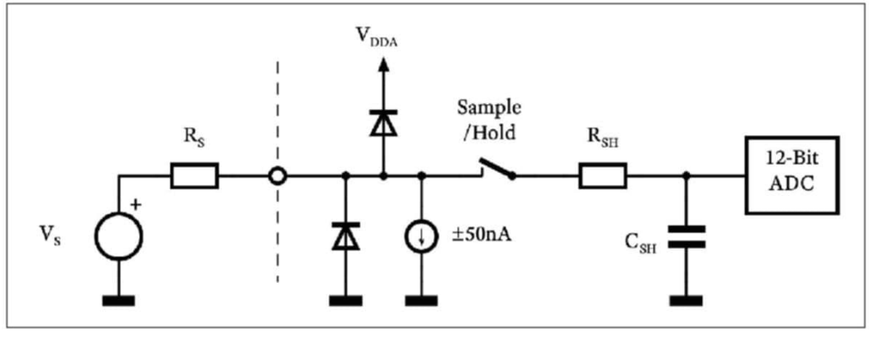


FIGURE 4
We need to add a buffer between the divider and the ADC input, to avoid errors caused by the ADC input leakage current, and to ensure the sample-and-hold capacitor charges fully.

on the sampling time. The STM32L03 data [1] indicates it should be below 500Ω for the fastest possible sampling time, and below 50kΩ for the slowest (at full clock frequency).

We therefore have two good reasons to insert a buffer amplifier between the divider and the ADC. A buffer is an amplifier with a gain of unity, but with a high input impedance and a low output impedance, and is easily implemented using an operational amplifier.

OP-AMPS

Operational amplifiers (op amps) are the bread and butter of the analog designer. They are so called because they can be used to perform a whole series of "operations," such as addition and subtraction, filtering, integration and differentiation, and a range of non-linear functions. I will cover some of this in further articles in this series, but for now we will concentrate on the simplest implementation—the non-inverting buffer.

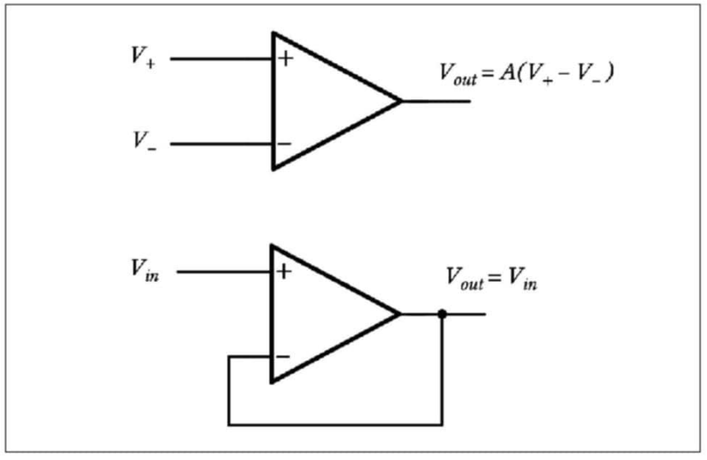


FIGURE 5
It is convenient to think of op amps as ideal components when analyzing the operation of circuits like this buffer. But we need to be aware of their many potential sources of error, in order to design reliable circuits.

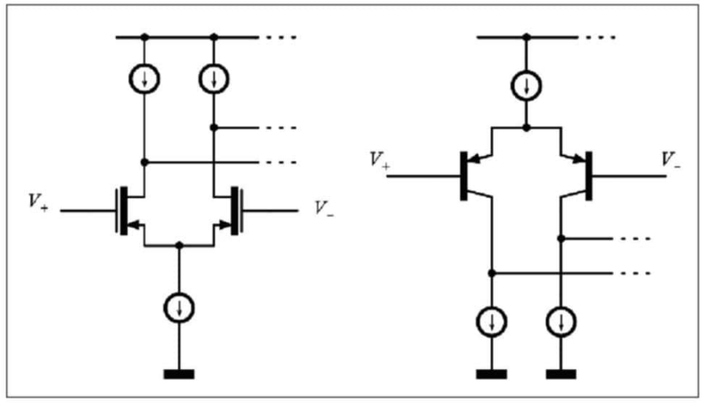


FIGURE 6
These simplified diagrams of op amp inputs help illustrate some of the non-ideal characteristics of op amp inputs. They include input bias current, input offset current, input offset voltage, and common-mode voltage range.

It is often convenient to think of op amps as ideal devices. Consider the op amp at the top of **Figure 5**. If it is ideal, the output voltage will be the difference between the two input voltages, multiplied by the open-loop gain (A), which will be infinitely large. If the two input pins are at the same voltage, the output will be zero. No current will flow into or out of the inputs, since they have infinitely high impedance. The output will have zero source impedance and will be able to swing to any voltage that may be required.

If we connect the inverting input to the output of the amplifier, as shown at the bottom of Figure 5, we provide negative feedback, and something almost magical happens. If V_{out} goes higher than V_{in}, the difference between input pins would be negative, and V_{out} would be driven lower. Conversely, if V_{out} is lower than V_{in}, the output will be driven high. The only stable situation is when V_{out} and V_{in} are equal. So, we have created a buffer with infinite input impedance, zero output impedance and a gain of exactly 1. I'll just put in my order for some ideal op amps…

Input Non-Idealities: Op amp inputs are usually composed of FET (JFET or CMOS) or bipolar differential pairs as shown in (very) simplified form in **Figure 6**. Because real op amps are built from real transistors, there must be some level of current flowing into or out of the input terminals. It will be much less in the case of the FET op amp, but it will not be zero. This is known as "input bias current" (I_B).

In real op amps, input bias currents vary considerably, from tens or hundreds of nanoamps in bipolar op amps to femtoamps (10^{-15}A) in the very best FET amps. Input bias current changes with temperature, sometimes by a surprising amount. As an example, the input bias current of a TL074H jellybean FET amplifier [2] is specified to be ±120pA at 25°C, but will vary as much as ±5nA over the full device temperature range. That's 42 times more than the nominal figure!

Manufacturing variances mean the input bias current will not be exactly the same on both pins of the same op amp. In fact the input bias current number you see in the data sheets is the average of the two. The difference between the two input bias currents is known as the "input offset current" (I_{OS}). If the circuits connected to each input of your op amp are identical, this is the figure you will be interested in. If they are asymmetric, like our buffer, input bias current may be more relevant.

Manufacturing variances produce another asymmetry in op amp inputs—"input offset voltage" (V_{OS}). This arises from the difference in transistor characteristics, such that one transistor turns on at a slightly different voltage than the other. This means that even with the input pins shorted together, the output of an open-loop op amp will saturate at one supply rail or the other. Input offset voltage also changes significantly with temperature. Input offset voltages of typical modern op amps range from several millivolts down to single digit microvolts in the best performing devices.

The final non-ideality of op amp inputs is also apparent from Figure 6. Under negative feedback, the voltage at both inputs is nearly identical, but the range over which they can operate is limited. This is known as the "common-mode voltage range" (V_{CM}). In the FET example on the left, V_{CM} could theoretically extend to the positive supply, but it can't reach quite to the negative supply, because the gate voltage of the FETs must be higher than their source voltages for them to be biased on. Similarly, in the BJT example on the right, V_{CM} cannot extend all the way to the positive supply, since the base-emitter junctions of the transistors must remain forward biased. Op amps designed for single-supply operation often have this latter input configuration, regardless of transistor type, since the negative rail is notionally 0V, and it is highly desirable to be able to connect an op amp input to zero.

These days there are many "rail-to-rail input" op amps on the market, where V_{CM} extends to both rails. These usually have two differential input stages, one of each polarity. The op amp switches from one stage to another depending on the level of V_{CM}. This works well for most applications, but it does come with some quirks. The input offset voltage and bias current can change suddenly when the stages switch, so read the data sheet closely to see if this will affect your design.

Selecting An Op Amp: The upper part of **Figure 7** shows a useful model of the input of an op amp that we can use to understand and quantify the errors it will produce. The input offset voltage is represented by a voltage source in series with one of the inputs, and the input bias currents are represented by current sources. In the lower part of the figure, I have applied this model to our signal conditioning circuit, with 0V input. The input bias current on the inverting input can be ignored, because the op amp's output can easily overcome this small current.

However, the input bias current and the input offset voltage at the non-inverting input will each contribute to a voltage error. The input bias current flows through the parallel combination of R1 and R2 (about 500kΩ), producing an error voltage. This adds to the input offset voltage error, is multiplied through the closed-loop gain (1 in this case), and appears at the op amp's output.

To get an idea of how big this error could be, we will take a look at the datasheet for a very basic but super low-cost jellybean op amp, the LM324 [3]. These are single supply op amps that work down to 3V. Input offset voltage is specified to be ±3.0mV (max) at 25°C. The data suggests that on top of this, V_{OS} will change ±7μV for every degree of temperature change, so we can expect a worst case offset voltage of ±3.1mV (±0.1% of full scale).

The LM324's input bias current is specified to be −35nA. The negative sign indicates that the bias current flows into the pin, since the convention is that positive current flows out of pins. With a 500kΩ source impedance, this corresponds to a whopping 17.5mV (±0.6%) error. To add insult to injury, the LM324's V_{CM} is specified to range from the negative supply up to 1.5V below the positive supply. With our +3.3V power supply this limits the input voltage to 1.8V. We will need to use a need a better op amp.

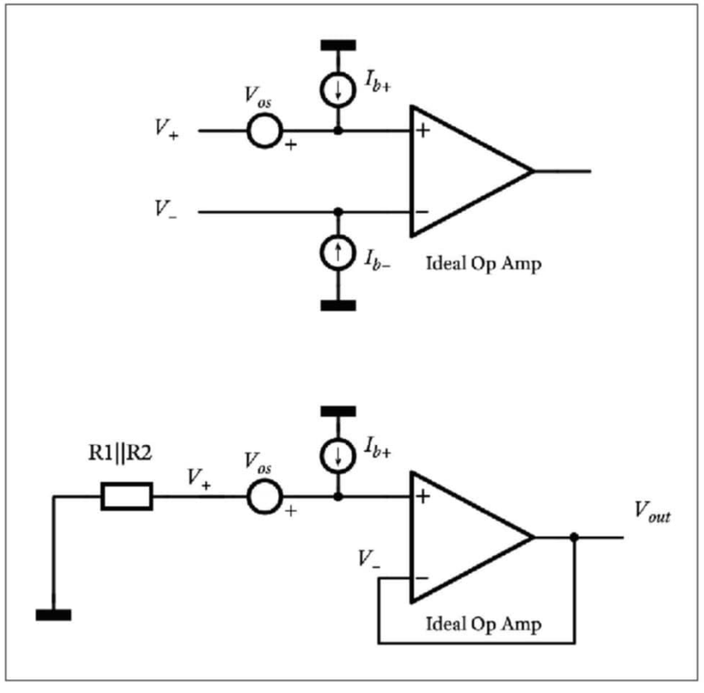


FIGURE 7
The upper diagram shows a useful model for op amp inputs. Below is the model applied to our non-inverting buffer and input attenuator. Input bias current and input offset voltage will each contribute to a temperature-dependent voltage error.

FEATURES

There are plenty to choose from, but we can narrow down the search by establishing some criteria; First, we need single 3.3V supply operation, with V_{CM} extending from the negative rail to within say 0.2V of the positive rail. Second, ideally, we don't want to have to trim out offset, so we want V_{OS} well under 1mV over the entire temperature range. Third, we'd also like the error due to bias current in the same range, so we need I_B to be under a nanoamp. Armed with some criteria like these, we can use one of the many on-line parametric search tools to find some suitable candidates.

I ended up choosing the TP553x family of op amps from 3Peak [4]. These are low-cost precision "zero-drift" op amps. Zero-drift op amps use a clever internal technique to minimise input offset voltage and its drift with temperature. A second internal "nulling" op amp alternately measures its own offset voltage and that of the main amplifier, nulling them both out. This process typically occurs many thousands of times per second, resulting in extremely low levels of offset and drift.

In the case of the TP553x family the maximum offset voltage is ±20µV with ±50nV/°C drift. Input bias current is 50pA ±7.5pA/°C. The input common mode voltage extends 0.1V beyond both supply rails. These op amps are naturally more expensive than the LM324, but at around $2 for a quad pack are represent very good value.

Op Amp Output Error: This is still not the complete story as far as the non-ideal nature of op amps is concerned, since we have only looked at the input side. The output stage of an op amp is usually a push-pull style output, with one transistor driving the output toward the positive rail, and one of driving it down toward the negative rail. No matter how hard they are driven, these transistors must have some minimum saturation voltage that prevents the output voltage quite reaching the supply rails.

Some op amps can get very close, like the TP553x we have chosen. This can swing to within 25mV of the rail (worst case) when driving a 10kΩ load. For our application this does not matter for the positive swing, since we are only expecting a little over 3.0V at full scale, and we have a 3.3V supply. However it does have serious implications for the bottom end of the range. If the voltage applied to the ADC can never be lower than 25mV, we won't be able to measure voltages below about 100mV at the input of the divider.

KEEPING AWAY FROM THE RAILS

This brings us to another analog design rule of thumb—try to avoid the span of critical signals getting too close to the supply rails. This allows you to avoid a lot of "drive-to-the-rail" problems and various non-linearities that occur as signals run into the power supply limits.

One way analog designers often address this problem is to use split power supplies. For example, if we powered our op amp from ±3.3V supplies, the problem disappears, since 0V is no longer aligned with one of the supply rails. Because split supplies are not available to us in this example, we will resort to another trick.

As shown in **Figure 8,** R1 and R2 attenuate the input signal as before, but now the bottom end of the divider (the negative input pin) is held to some fixed voltage by the buffer U3. If R3 and R4 are chosen, so that this offset voltage is about 150mV, the voltage at the output of the attenuator (and therefore that applied to ADC Channel 1) will be 0.15V when V_{in} is 0V, and around 3.15V when V_{in} is 12V. We have effectively shifted the span of the input signal such that it extends from 150mV above the zero-volt rail to 150mV below the 3.3V rail.

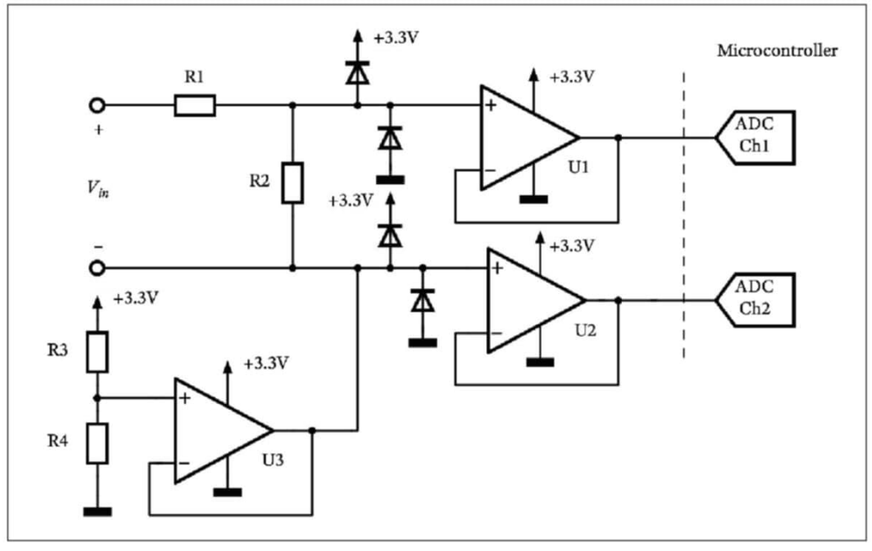


FIGURE 8
By shifting the bottom of the divider up by 150mV via U3, we avoid the span of our signal running into either of the supply rails and the associated problems. The offset voltage is subtracted from the signal inside the microcontroller.

Because the signal is kept away from either supply rail, we neatly avoid the op amp output swing problem and the very ends of the ADC conversion span, where there may be some non-linearity. We have, however, added an offset to our signal that will influence the measurement.

This is where the buffer formed by U2 and the second ADC channel come in. The offset voltage is also digitized, so it can be subtracted from the measurement in firmware. The result is effectively a measurement of the voltage across R2, which is proportional only to the input voltage we are measuring.

The specifications for R3, R4, and U3 don't have to be particularly precise, and can even have meaningful temperature drift, since the offset voltage is nulled out of each measurement as it is made. For best results, U1 and U2 should be identical and preferably in the same package.

The circuit of Figure 8 has another trick up its sleeve. If you set the values of R3 and R4 equal to each other, the resulting offset would be about 1.65V, and the circuit would be capable of measuring an input voltage range of ±6V. In fact, I have used this circuit in an application where the measurement could be switched between bipolar and single-ended in different operating modes.

Figure 8 also shows how we can protect this circuit from high voltages. The four diodes prevent the op amp inputs from exceeding the positive or negative supply rails by more than one diode drop (about 0.6V). The high values of R1 and R2 means that the maximum current through these diodes will only be about 50µA when 100V is applied to the input.

The op amps have similar internal protection diodes that might suffice, but it's a good idea to add some external diodes to prevent them from being overstressed. For a "belt-and-braces" approach, you could add a resistor (for example, 1kΩ or thereabouts) between the external diodes and the op amp input pins to further limit the current through the internal diodes.

KEEPING TRACK OF ERRORS

That's about it for our design! What sort of performance can we expect? I find it handy to use a spreadsheet to keep track of the errors in a circuit such as this one. It does not have to be anything complicated, as **Table 1**

	Error	Nominal Value	At Nominal 25°C		Over 10 – 40°C (Nominal ±15°C)	
			Absolute Error	Relative Error	Absolute Error	Relative Error
1	R1/R2 Divider (Susumu RGT 2MΩ /680kΩ, 0.1%, 10ppm/°C)	253.7E-03	380.6E-06	0.150%	418.7E-06	0.165%
2	Divider output (12V input span)	3.045E+00	4.6E-03	0.150%	5.0E-03	0.165%
3	Buffer Input Offset Error (TP553x, 20µV, 50nV/°C)	000.0E+00	20.0E-06	0.001%	20.8E-06	0.001%
4	Buffer Bias Current Error (TP553x, 50pA, 7.5pA/°C, 500kΩ)	000.0E+00	25.0E-06	0.001%	81.3E-06	0.003%
5	Total Untrimmed Voltage Error at Buffer Output (Lines 2+3+4)	000.0E+00	4.6E-03	0.151%	5.1E-03	0.168%
6	Total Trimmed Voltage Error at Buffer Output (0.2 x Line 2 + Lines 3 + Line 4)	000.0E+00	958.4E-06	0.031%	1.5E-03	0.049%
7	ADC Error (STM32L03 MCU, 11.3 ENOB with 256 x Oversampling)	3.30E+00	1.3E-03	0.040%	1.3E-03	0.040%
8	Untrimmed Voltage Ref Error (STM32L03 MCU, 1.22V ±5mV, 100ppm/°C)	1.22E+00	5.0E-03	0.408%	6.8E-03	0.558%
9	Trimmed Voltage Ref Error (STM32L03 MCU, 1.22V ±0.5mV x , 100ppm/°C)	1.22E+00	500.0E-06	0.041%	2.3E-03	0.191%
10	Total Untrimmed Error (Line 5 x Line 7 x Line 8) at 12V full scale	000.00E+00	72.0E-03	0.600%	92.0E-03	0.767%
11	Total Trimmed Error (Line 6 x Line 7 x Line 9) at 12V full scale	000.00E+00	13.4E-03	0.112%	33.6E-03	0.280%

TABLE 1
Example spreadsheet for keeping track of the errors in a circuit. It helps you to see where problems are, and to evaluate potential improvements.

attests. The first row shows the error due to the voltage divider, while the second shows the error in the nominal, full-scale output due to the divider uncertainty. Rows 3 and 4 are the buffer op amp input offset and bias errors, respectively. Row 5 is the total untrimmed error we could expect to see at the ADC input.

I calculated the relative and absolute errors at 25°C and over the operating temperature range. It is obvious from Row 5 that the 0.15% uncertainty due to the divider resistance tolerance dominates the overall analog error. Row 6 shows the resulting error, if we could null out just 80% of the divider error. This would reduce the analog error to less than ±0.05%.

Other Errors: I have included the error due to the ADC on Row 7. I will cover this topic in some detail in a later article, but for now, suffice it to say that the 12-bit ADC has about 10 effective bits of precision due to its offset, gain, and non-linearity errors. This can be improved to an effective number of bits (ENOB) of 11.3 when using an oversampling factor of 256 times. This corresponds to an error of about 0.04%, bringing the total analog plus digital error up to the 0.1% range.

This seems pretty good, but we also have to include the error in the ADC's reference voltage. This is a band-gap reference, which is specified to be accurate to ±5mV at 25°C with 100ppm/°C tempco, corresponding to almost 0.6% error (Row 8) over the operating temperature range—by far the dominant source of error in our system.

Fortunately, the fixed part of the reference error will also be trimmed out by the calibration process, as shown in Row 9. Of course this does nothing to ameliorate the relatively high temperature drift error of this reference. The final result is that, with calibration, we could expect a worst-case error of around 0.1% at room temperature, and 0.3% over the 10°C to 40°C temperature range (Row 11). If we wanted to improve this further, using an external voltage reference with a better tempco would be the place to start. With a better reference, it should be possible to get 0.1% precision error out of this circuit across the temperature range. This is the advantage of building an error spreadsheet like Table 1 – it is easy to see where the problems lie and to evaluate potential improvements.

As mentioned at the start of this article, I aim to give real-life examples, so I actually built the circuit shown in Figure 8. The results showed a worst case uncalibrated error of around 0.3% at room temperature (about 20°C on the day I made the measurements). This is considerably better than the estimated untrimmed error of 0.6% shown in Row 10 of Table 1 – not a surprise, given the statistical nature of these errors.

OPTIMIST OR PESSIMIST

So, does the approach of adding worst-case errors provide an overly pessimistic view of the performance of circuits like this? If we have five or six (or more) different sources of error, what are the odds that they will all align in the worst possible way?

I think this is exactly the right question to ask. The errors we are speaking of are statistical in nature and usually Gaussian in distribution. The chance of being at the extremes is way lower than the chance of being near the middle of the range. So the amount you should worry about this stuff depends on the level of risk involved, and the consequences of failure to meet specifications.

On the one hand, if the consequences are serious—if lives are at stake, mistakes are expensive to fix, or your brand's reputation is on the line—you might want to err on the side of caution, even if the chances of failure is remote. Similarly, if you are making hundreds of thousands or millions of something, the odds are high that there will be outliers within the population, so it makes sense to be conservative, even if the consequences of a single failure is modest.

On the other hand, if you are building a small number of high-value items or a project just for yourself, you may elect to take your chances and switch out components if you are unlucky enough to run across an outlier. There is no right answer, but the key is to be aware of the potential sources of error, make the appropriate calculations and make informed decisions.

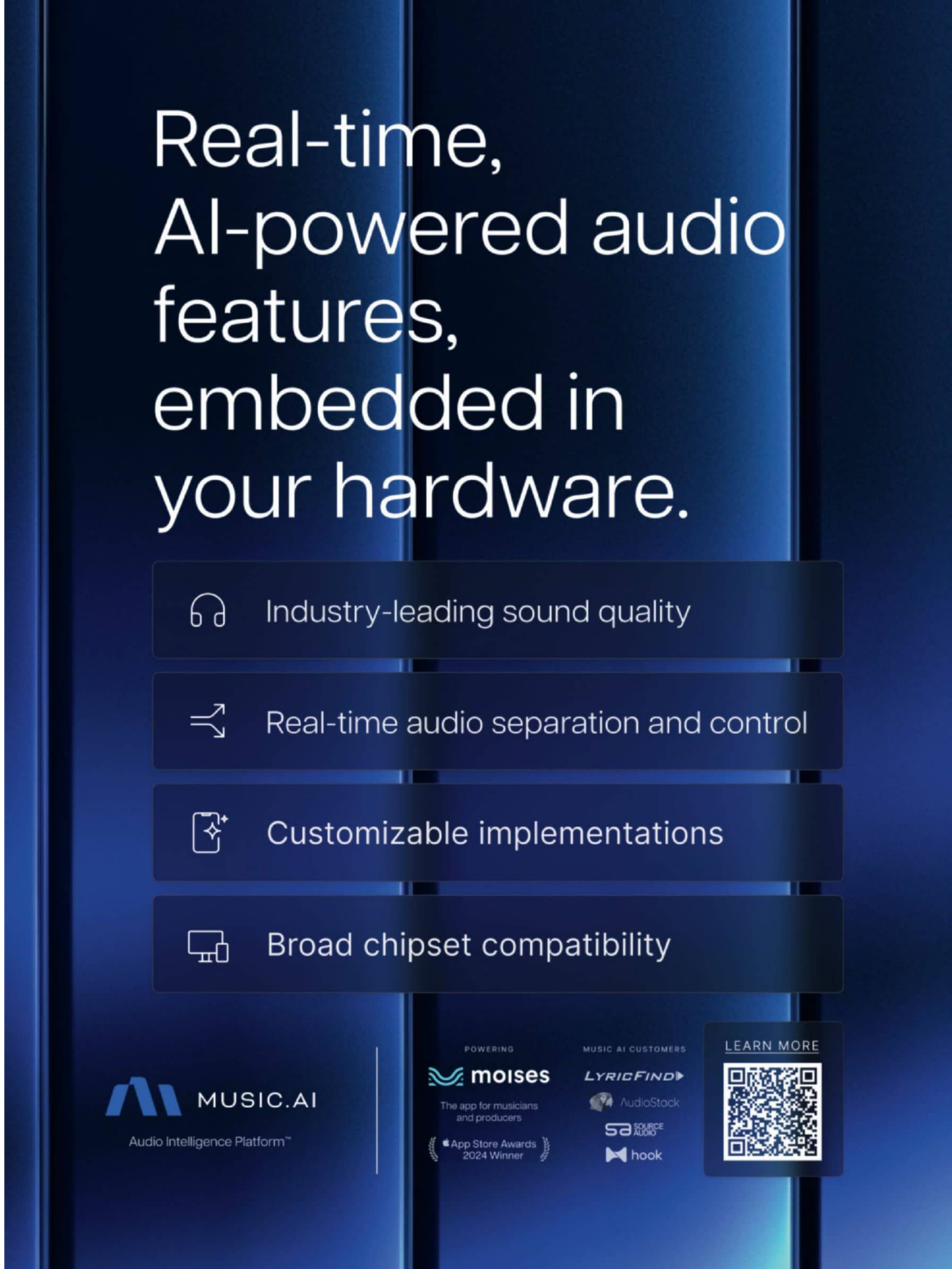
In Part 2 of this article series, I will take a deeper dive into op amps and explore some useful circuits that all analog designer should have in their toolkits.

ABOUT THE AUTHOR

Andrew Levido, (andrew.levido@gmail.com) earned a Bachelor's degree in electrical engineering in Sydney, Australia. He worked for many years in R&D and management roles for power electronics, telecommunication, and IT companies. Andrew has maintained a hands-on interest in electronics, particularly embedded systems, power electronics, and precision analog design. Over the years, he has written many technical and educational articles for various publications. He also provides consulting services to a number of clients.

Additional materials from the author are available at: www.circuitcellar.com/article-materials

References [1] to [4] as marked in the article can be found there.



Keeping Time with Rolling Balls

An Arduino-Controlled Gravitram Clock

By
Aubrey Kagan

An Arduino Zero turns out to be the perfect controller when Kagan sets out to keep time with stacked columns of rolling balls in a novel desktop clock

At the onset of my retirement in 2021, a friend challenged me to design a control board to make an antique clock keep time accurately, without modifying mechanical pieces of the clock. I described the project in my article, "Antique Clock Accuracy Project: Reloaded" (*Circuit Cellar* 381, April 2022) [1].

After I completed the project, I cast around for a new one, and my thoughts continued in a timekeeping vein.

Although the word "gravitram" may not be familiar, you are certainly familiar with the object. Also known as a "rolling ball sculpture" or "marble run," it consists of a ball running through a series of twists and turns under the influence of gravity, often with a Heath Robinson ethos. Many years ago, a cousin had a clock of this nature—except it only had a single electric motor and mechanical interlocks. I thought I would use the concept for my own clock.

The first hurdle was how to raise the ball to the start of the run. I tried a primitive rail gun to launch the metal ball without success. I tried using a large solenoid to launch the ball, but I couldn't get enough height. Then I investigated a robotic arm, and I found one with sufficient height (460mm). The "wow" factor for my grandchildren led me to opt for this approach.

In order to present the time, I wanted four vertical columns (marked B in **Figure 1**), where the number of balls in each tube would indicate a digit of the time. Rather than create open channels for the balls, I decided on PETG transparent tubing. It is easy enough to shape the tube with a heat gun—together with a silicone bending insert and a bending kit—although I did occasionally shape the tube freehand [2].

The next order of business was to direct the balls to a particular column, and it seemed logical that the arm would have a separate entry point for each column (marked A in Figure 1). I wanted a tall device where the balls would have a spectacular, winding descent. Since the arm couldn't reach high enough, I ended up with a slanted clock face (marked C in Figure 1) and the stacked feeder tubes you can see in the photo.

The tubes are attached to the wooden frame using transparent acrylic shapes I designed using QCAD and were cut out of 12mm (1/2") acrylic sheet on a laser cutter. The connecting tubes and funnels were designed using FreedCAD and 3D printed. I have access to those tools through my membership in Ylab, a local maker space.

ELECTRONIC HARDWARE

The gravitram is controlled by an Arduino Zero, with an Adafruit 16-Channel 12-bit PWM/Servo Shield together with an Adafruit Clock and Data Logging shield [3]. The Arduino Zero was chosen specifically because it has an embedded debugger, which I found to be essential to developing the project.

I wrote a user manual for the Arduino Debugger in my article, "Evaluation of Debugger on Arduino IDE V2.0.4" (*Circuit Cellar* 397, August, 2023 [4]. I must admit to being disappointed that there does not appear to have been any revisions to the Arduino debugger since my original evaluation.

The digital outputs of the Arduino are buffered by an octal Darlington buffer, the ULN2803, and I used an opto-isolator to isolate a switch input that was causing anomalous operation of the micro.

All inductive loads have back-EMF diodes.

THE ROBOT ARM

I managed to assemble the arm after finally getting the assembly instructions and many hours of running the unit [5], then found there are some "gotchas" that you need to consider. For example, the vibrations cause the screws to loosen. Before you break out the thread locker, you should know that if you need to replace a servomotor, you need to disassemble the arm in reverse order to the assembly. Rather than thread locker, use silicon glue (which can be peeled off) or M3 Nyloc nuts.

There are two cantilevers on the base structure of the arm (marked A and B in **Figure 2**), and they contribute to a significant wobble. I used a longer bolt for the horizontal mounting to steady the y-dimension wobble (marked C in Figure 2) and an additional bolt to steady the x-dimension wobble (marked D in

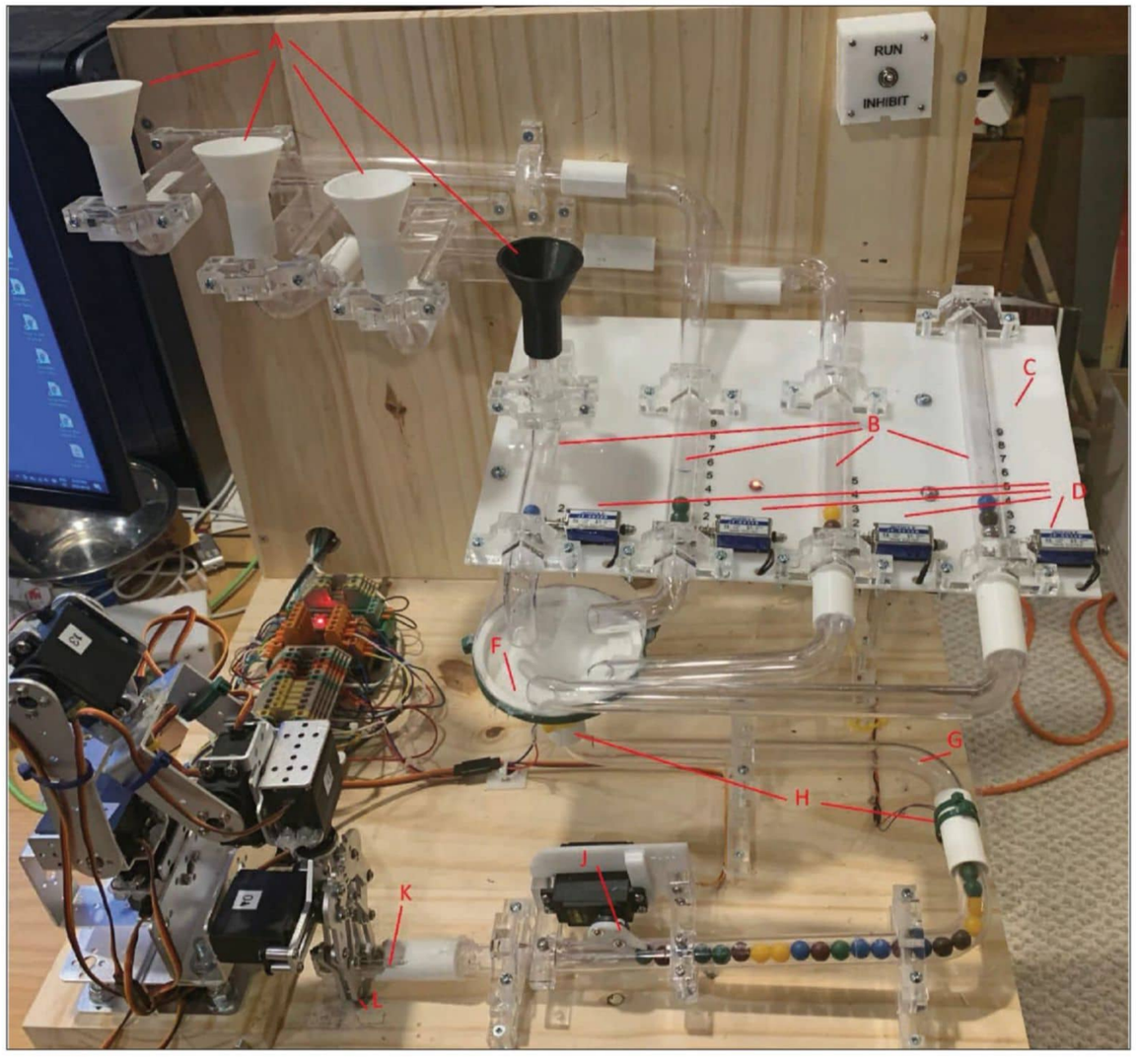


FIGURE 1
Gravitram with the robot arm in the home position. Key: A—entry funnels; B—time tubes; C—clock face; D—solenoids; F—final funnel; G—feed tube; H—vibrators; J—pawl; K—landing strip; L—home position of claw.

Unrelated to the wobble, I added spaced washers (marked E in Figure 2) to enhance the grasp of a ball with the claw.

The claw itself has a rather primitive gear, as you can see in **Figure 3**. Take note that in gripping the ball, there is only one point of contact between the two gears (marked A in Figure 3), and should the claw accidentally bump into something (which happens a lot in testing), the claw gears can skip a tooth, confounding all the settings that have been tested.

The arm is designed to use the MG995R and MG996R servo motors—the difference being plastic versus metal gears—since the length and weight of the arm segments will add to the strain on the gears. I opted for the latter.

SERVO MOTORS

Hobbyist servo motors like the MG996 have the servo feedback built into the motor and are controlled by a pulse width modulation (PWM) signal [6]. The servo moves to and then holds its position depending on the PWM signal.

The motor will rotate through approximately 180º as the PWM signal goes from 0% to 100%. Power to the motor derives directly from the power supply, making the design of the drive of the PWM signal trivial, driven directly from 5V logic. The PWM signal is part of a 1ms waveform in a 20ms period. Some texts describe the control as 1.5ms to 1ms as turning the motor counterclockwise and from 1.5ms to 2ms as causing a clockwise motion. I prefer the alternative view—that the PWM signal determines the angle from 0º to 180º, which allows me to predict which way the arm joint will move in response to a PWM change.

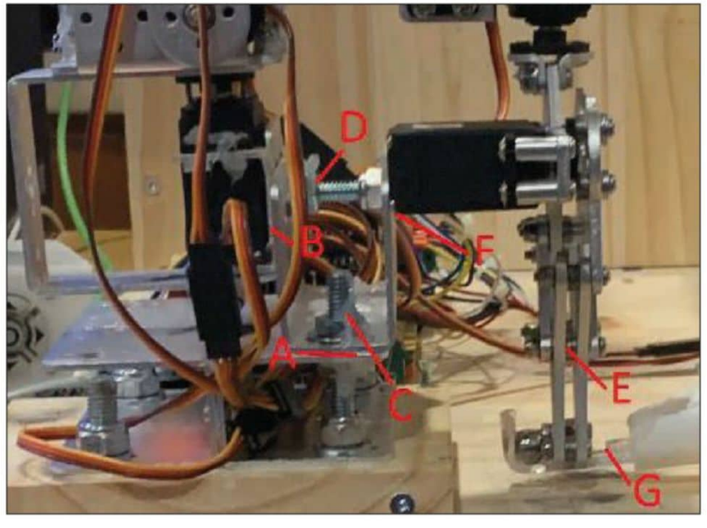


FIGURE 2
Claw in start position. Key: A—horizontal cantilever; B—vertical cantilever; C—steadying bolt for A; D—steadying bolt for B; E—spacer washers; F—stall position; G—landing strip

Starting for the first time is particularly vexing. You have no way of knowing how the motor gear is oriented. The coupling disk that the arm attaches to can also be mounted on the motor gear in many different positions, so you must test as you mount each motor onto the assembly.

Each servo motor responds with alacrity; given the inertia of the robotic arm, the sudden movement can be problematic, especially during development, as it crashes into the world around it. When the power is cut, the motors cease maintaining their position and the whole structure collapses.

I found that it was best to position the arm to its rest position, marked L in Figure 1, which was very close to its power up position—so there would be no large-scale sudden jerk when power was re-applied—and then cut the power. At the very least, be prepared to catch the arm if you need to disconnect while the arm is in the air.

The movement of the arm is choreographed through several positions: on the way from ball pickup, to the respective entry funnel, and return. To prevent rapid movements, for each servo motor the algorithm takes the current setting and increases it by one every 10ms until it reaches the target setting.

A 12-bit PWM has a resolution of 4096 steps, which covers the full 20ms range used by the servomotor. However, the 1ms operating range means that the resolution is reduced to 204. Across 180º, that translates to a resolution of 0.88º. On average, a segment of the arm is about 110mm long, i.e. a radius of 110mm of a circle that it circumscribes.

Basic geometry yields a circumference of about 691mm, and this means that there is an uncertainty of the position of the end of each joint of ((0.88/360)*691)=1.7mm. That is without any of the other issues like play in the gears, resolution of the internal circuitry of the servomotor, PID tuning, and the internal controller deadband.

Across several joints, the errors can accumulate to make accuracy problematic. The problem was particularly noticeable in picking up the ball.

A few different approaches could have helped with this. Adafruit sells a D125F servomotor with the wiper of the feedback potentiometer brought out on a separate wire [7]. I was unconvinced that this could solve my problem, and I discounted it since it is only available with plastic gears, and I really couldn't face up to another 6 wires that could potentially entangle the arm.

I also considered the DYNAMIXEL Motor - AX-12A, which has a serial port connection rather than a PWM interface [8], but it would require mechanical acrobatics to make it fit.

In the end, I gingerly experimented with—and succeeded in—driving the arm into a stall position against the base, mechanically limiting its motion (marked F in Figure 2).

The six servomotors together use quite a bit of power. Each draws between 500mA and 900mA at 5V and can go up to 2.5A in a stall. Since the arm only works for about 15 seconds every minute,

I chose to move the arm to its rest position and turn off its power for the remaining 45 seconds. The 5V power to the arm is switched through a solid-state DC relay.

POWER SUPPLIES

The availability of certain parts meant that it was easier to add a power supply than to find a part with a suitable voltage rating. As a result, there are three mains-sourced power supplies: 5VDC, 12VDC, and 24VDC.

SOLENOIDS

In my 30+ years of experience I have bandied the word "solenoid" around with much aplomb, until I came to order some for my clock. In all these years, it never occurred to me that the armature (the moving metal slug) would come out unless held in by gravity or some mechanical method, like a nut on one side.

It took 3 purchase attempts to determine that I wanted a "push-pull" solenoid [9]. One disadvantage is that there must be clearance at the back of the solenoid.

Each time-tube has a solenoid armature projecting through the tube, trapping the balls above it and indicating the particular digit of the time as marked D in Figure 1. When the system calls for the digit to be set to zero, the solenoid is energized for a short period— long enough for the balls to exit and run down the outlet tube to the final funnel (marked F in Figure 1). To prevent overfilling the final funnel, when more than one-time tube is to be zeroed, the solenoids are activated sequentially.

VIBRATORS

In some circumstances, several balls would choke the throat of the final funnel. I found that adding a small vibrator (marked H in Figure 1) solved the problem [10].

It took 6 hours to 3D print the funnel, and I am not sure if there is any design that could guarantee the blockage would not happen, so I was satisfied despite the extra hardware required.

The balls from the final funnel roll into the feed tube (marked G in Figure 1) and around the U bend. The slope of the tube just before the first bend is not quite sufficient to start the ball rolling if the ball is stopped by the queue ahead of it. A second vibrator took care of that.

The vibrators make an annoying buzz, so I only turn them on when the solenoids are activated and when the pawl (more on tht in a moment) passes a ball to the landing strip.

LANDING STRIP

I termed the zone where the ball is grasped by the claw the "landing strip" (Figure 1k and, for a better view, Figure 2g). The tube has been cut away, leaving a narrow channel so that the ball is almost completely exposed and accessible to the claw.

I started the design using a line of steel balls, but the impact of each new ball would sometimes dislodge a ball. I changed to plastic balls, which reduced the problem but did not solve it.

As an aside, it is possible to dye plastic (depending on the type) allowing for a festive display of colored balls in presenting the time!

PAWL

I decided that the only solution to the ball displacement was to add a pawl (marked Jin Figure 1 and **Figure 4**). It is connected to a servomotor and driven in the same way as the servomotors in the arm. The pawl tees up another ball just after the claw has gripped a ball and begun its motion toward the entry funnels.

OPERATION

To allow concurrent activities, I implemented a co-operative multitasking approach in software.

On initialization, the processor reads the time from the on-board clock of the Clock and Data Logging shield. To ensure there is a single ball on the landing strip, the arm fetches a ball and drops it to the tens of hours tube.

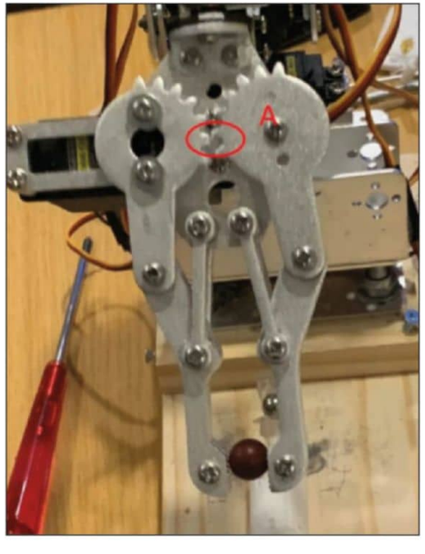


FIGURE 3
The gear mesh on the claw while gripping a ball.

FEATURES

At the same time, the pawl delivers the next ball. Then all the time-tubes are cleared sequentially since the processor has no knowledge of the contents of the tubes, and this will initialize them all to zero.

Because it takes over 8 minutes to pre-load all the balls to their respective tubes in the worst case, 9 minutes is added to the current time and that advanced time is set up in the tubes. The processor then waits till the actual time reaches the advanced time and then proceeds to update the time every minute.

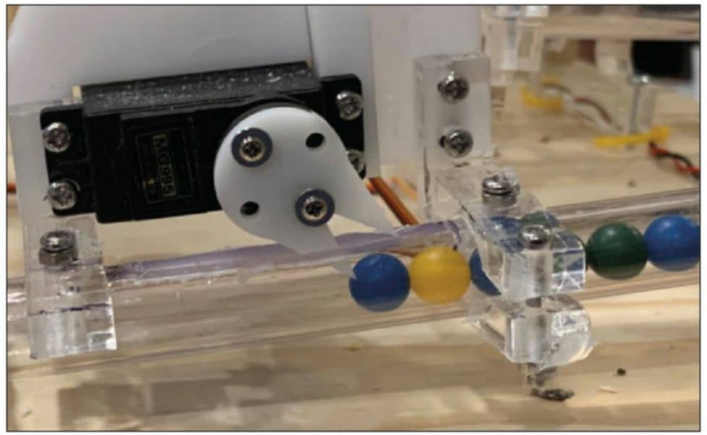


FIGURE 4
The Pawl. The servomotor moves slowly clockwise, gripping a single ball in the inverse V until the ball is released to roll towards the landing strip. The trailing part of the V prevents the next ball from following it. The servomotor then turns counterclockwise, pushing the whole line of balls back up the tube until the next ball enters the V.

ABOUT THE AUTHOR

Aubrey Kagan has worked in electronics for more years than he cares to remember. He is now retired and happy to give advice where he can. He has written many articles for *Circuit Cellar* over the past 25 years as well as a book, *Excel by Example,* based on three of those articles. Aubrey was one of the "notable contributors" interviewed in *Circuit Cellar*'s 25th Anniversary issue. He has also published several design ideas as well as numerous blogs covering many aspects of electronic design. You can find a list and links to more of his publications on the *Circuit Cellar* article materials webpage.
He can be contacted at akagan@sympatico.ca.

Additional materials from the author are available at:
www.circuitcellar.com/article-materials

References [1] to [12] as marked in the article can be found there.

RESOURCES

QCAD | www.qcad.org/en

FreeCAD | www.freecad.org

I found it interesting that at every minute update, no matter how many tubes are zeroed (by activating the solenoids) only a single ball is ever added.

The Arduino only looks for a change of minutes, which initiates the advance of the time. Since the time presented is open loop, the processor knows how many balls are supposed to be in each tube but has no means of verifying that.

Although the internal clock keeps 24-hour time, the displayed time on the tubes uses a 12-hour format.

Those of you with sharp eyes might have noticed an INHIBIT switch in the top right-hand of the system pictured in Figure 1. It does not stop the process immediately but allows the arm to park itself in the home position before stopping the clock.

Once inhibited, the time, as represented by the balls, will remain frozen until flipping the switch back to the RUN position. This requires that the clock re-initialize the time.

Because the clock is quite noisy, I added the ability to set a quiet time during which the operation of the clock is suspended, for instance, at nighttime. Again, the clock is re-initialized at the end of the quiet time.

SETTINGS

To set the time, I added a serial interface to a Matrix Orbital LK162B display and keyboard module (**Figure 5**), creating a hand-held terminal (HHT) [11]. The start and stop times of the quiet period is also set from this HHT. All of these times are set in 24-hour format.

Although the HHT can be disconnected without affecting the system operation, when it is present I use it to provide a monitor of the system operation.

CRASHES

I am still not convinced that aging and/or bumping the clock will not mean having to revisit the position settings at some time in the future.

Certainly, I have had several physical crashes when the Arduino went through figurative software crashes. It seems that the Arduino is susceptible to noise when long wire runs feed the digital inputs.

The fact the microcontroller works at 3.3V may be a contributing factor. One of the crashes actually stripped a motor gear, along with requiring a complete reworking of the choreographed points.

I added isolation wherever I could to hopefully prevent future crashes, but how do you prove that the problem is completely solved?

A watchdog timer wouldn't be much help since the reset would cut the power, and the arm would collapse, and the microcontroller wouldn't know the positions of each servomotor on the resulting reset. All of this gave me an added appreciation of the solutions required with a practical robotic arm.

On power up, it seems that a position sensor would help, but I imagine brakes are also necessary to hold the position when power is removed. But what happens when a microcontroller goes crazy and starts swinging the arm to unforeseen positions?

A microcontroller at each joint might limit the erroneous movement to only one joint. And then a current detector for the motor current since the current would increase when the motor is in a stall position.

FIGURE 5
Keyboard and display to change the operating parameters. The case is my design.

A POWER-HUNGRY BEAUTY

One of the shortcomings of the Arduino Zero is that it doesn't have separate electrically erasable programmable read-only memory (EEPROM). I have some concerns that the choreographed positions of the arm may need to change over time.

I am considering moving the array of the positions into a SD card on the Clock and Data Logging shield to act as non-volatile memory. I can then use the HHT to bump the settings as necessary.

I am also troubled by the dangling cables along the sides of the arm, and I am unsure how to affix them without them getting tangled and catching on the angular pieces of metal, as has happened in the past.

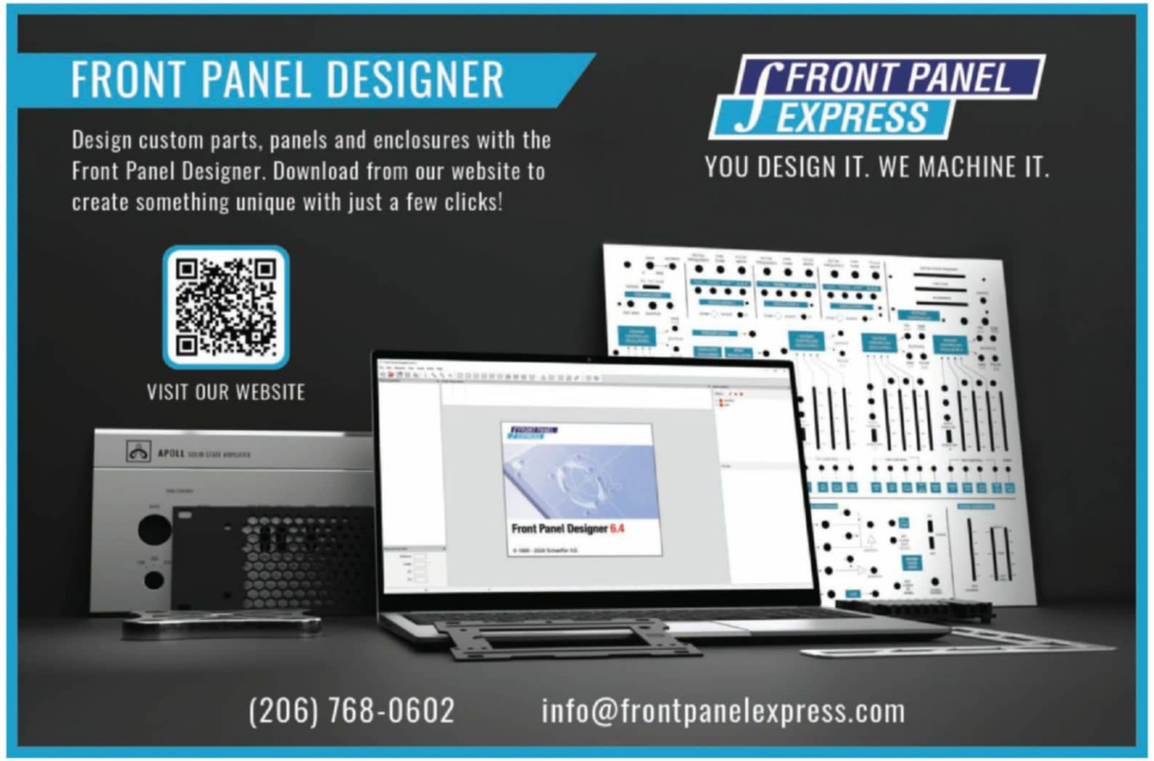
This project was developed ad-hoc, as you can probably see. I am still amazed I got it all to fit. Correcting the flaws would mean disassembling and reassembling a rather finicky project, and it is not something I am about to do. Let's say it adds whimsy!

I also had the intention of installing multi-color LEDs underneath the acrylic mounts, but once again that would mean disassembly. I did think of deriving the time from the Internet, but some of the places I want to demonstrate the clock are Internet-challenged, and at any rate, it would have required yet another shield.

My clock is probably as good as it is going to get!

You can see a video of my gravitram in operation on YouTube [12], or just search for "gravitram clock."

Everyone I know finds the stream of colored balls every 10 minutes entertaining, although this is probably the most power-hungry desktop clock ever. And it doesn't even go "tick-tock."


FRONT PANEL DESIGNER
Design custom parts, panels and enclosures with the Front Panel Designer. Download from our website to create something unique with just a few clicks!
FRONT PANEL EXPRESS
YOU DESIGN IT. WE MACHINE IT.
VISIT OUR WEBSITE
Front Panel Designer 6.4
(206) 768-0602
info@frontpanelexpress.com

Empowering Disabled Persons with Hand Motions for Environmental Controls

By
Matthew Oppenheim

Real-Time Sensor Display Using Python and a Smart Watch

Building on an earlier project about displaying real-time data, Matthew discusses how he used Python with the accelerometer of a Lilygo programmable smart watch equipped with an ESP32 microcontroller to enable an individual with cerebral palsy to operate environmental controls by using hand motions.

In this article, I explain how to use the Python programming language to parse and display accelerometer data from a smart watch in real time. The software can save the accelerometer data to a file. I used the software to record accelerometer data from a programmable smart watch, while recreating the hand motion of a person with cerebral palsy. By analysing the recorded data in a Jupyter notebook (a web-based, interactive computer platform) I was able to develop an algorithm to identify the hand motion. I ported the algorithm to the smart watch to enable the individual with cerebral palsy to operate environmental controls through hand motion.

I briefly talked about real-time display of sensor data using the PyQtGraph library in an earlier article I wrote for *Circuit Cellar* ("Full-stack Python." *Circuit Cellar* #319, February 2017) [1]. I have improved the code since then, and this article covers the project in greater detail. I created a YouTube video showing the software in action alongside the smart watch that produces the sensor data [2]. **Figure 1** is an example display from the software, with x, y, and z accelerometer data and two graphs created by processing the accelerometer data. All of these update in real time.

ONE SIZE DOESN'T FIT ALL

I use a Lilygo T-Watch S3 programmable smart watch [3] as a part of a system called "Handshake" [4]. Handshake enables people with severe physical disability to control software that creates speech, or to operate environmental controls by recognising an intentional hand, arm or leg movement. The T-Watch S3 contains a Bosch Sensortec BMA423 3-axis accelerometer and an Espressif ESP32-S3 MCU microcontroller, along with a smorgasbord of other goodies.

I wrote firmware for the ESP32-S3 that processes the accelerometer data to identify intentional hand motion. The ESP32-S3 then generates a radio trigger, which is received by a Lilygo T-Embed module that also contains an ESP32-S3 MCU. The receiver module sends a trigger to a device running communication software, using a standard switch controller interface.

The communication software is known as "switchable software," because it can be controlled with as little as a single switch or controller. For more information about the receiver unit and how it controls communication software, please see my article, "Create Your Own PCBs with a CNC Milling Machine," in *Circuit Cellar* #414, January 2025 [5]. For testing, I use Grid communication software by Smartbox [6]. This company kindly gave me a licence to use for this project.

I recently had successful real-world testing with the target user-group at the National Star College in Gloucester, UK, by using a simple algorithm for detecting intentional hand motion. However, this simple algorithm failed to work reliably with one of the participants. What to do? Obviously, there was no "one-size-fits-all" solution to identify the different types of hand motions made by the participants in my target user-group.

I needed to do two things: 1) analyze the sensor data generated by the T-Watch S3 worn by the individual for whom my algorithm did not work; and 2) based on this data, create a new algorithm to recognize the intentional motion.

How hard could that be?

RELIABLE MOTION

The Lilygo T-Watch S3 has a micro-USB socket that is used for loading code and charging the device (**Figure 2**). The same micro-USB socket is used to send and receive serial data from the ESP32-S3 contained in the watch. When I connect the watch to my PC using a USB cable, a new serial port appears. I use it to receive serial data from the watch.

The specialist college that I work with supplied a video showing the hand and arm motion of the participant who wanted to use the Handshake system. The algorithm I first implemented in the firmware on the watch had failed to identify that motion reliably. The problematic motion was a "slow-swinging" motion, whereas my initial algorithm was written to identify the "fast punchy" motion that successfully identified the intentional hand movement of several other participants in the target user-group.

A NEW ALGORITHM

I recreated the slow-swinging motion shown on the video, while streaming accelerometer data from the T-Watch S3 through the serial port. I wrote a Python script that runs on my PC to read and parse the accelerometer data from the serial port to recover the accelerometer x, y and z axis data. The data for each axis is displayed in real-time using the PyQtGraph library [7]. I use the PyQtGraph library, since it both displays graphs and can generate standard graphical user interface (GUI) widgets such as buttons. I added a button widget to my display to give the option of writing the accelerometer data to a text file.

I used a Jupyter notebook to analyse the accelerometer data that I recorded to file, while recreating the slow-swinging hand movement. To do this analysis, I modified some of the code that I wrote for the real-time display to display the saved data in the

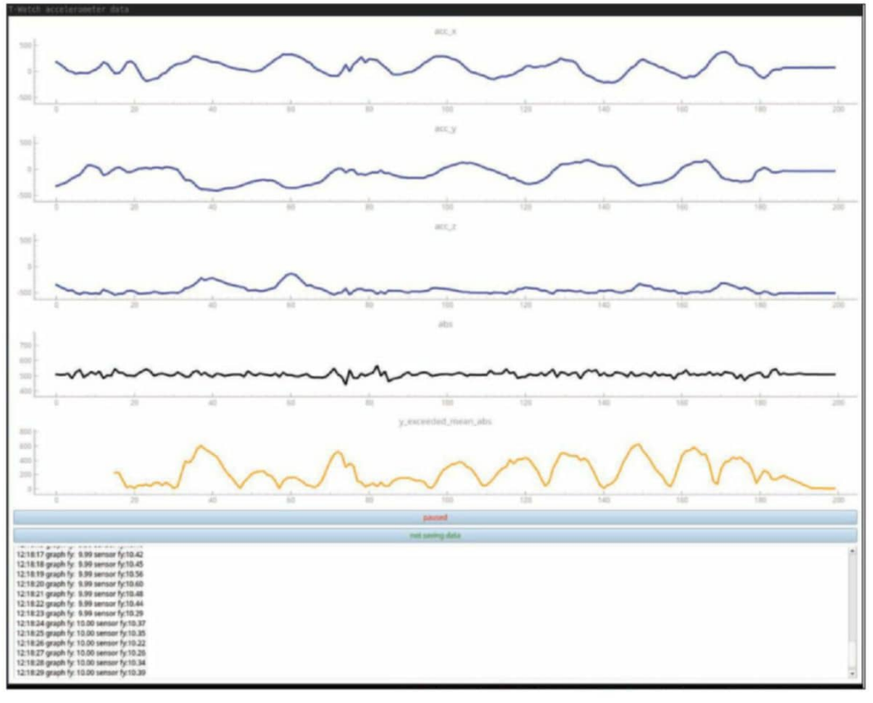


FIGURE 1
Real-time display of accelerometer and processed data from a Lilygo T-Watch S3.

Jupyter notebook. I leveraged the Pandas library [8] to apply different algorithms to the accelerometer data, and then graphed the results alongside the original data.

The simple algorithm that I'd written for detecting intentional movement did not clearly identify the slow-swinging motion. However, I could see that the y-axis accelerometer showed an identifiable pattern as I recreated the motion. **Figure 3** shows the raw x, y, and z accelerometer data, with a plot of the absolute acceleration created by processing the raw accelerometer data. The red box on this plot shows the area where I recreated the motion on which I want the `Handshake` system to trigger.

FIGURE 2
Lilygo T-Watch S3 with USB cable attached.

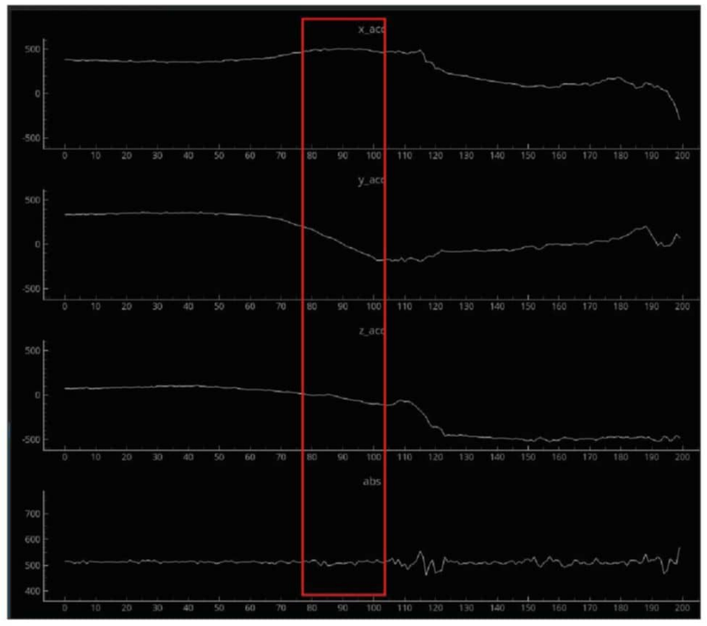


FIGURE 3
Real-time and processed accelerometer data from the T-Watch S3. Box indicates motion made by the participant.

The y-axis sensor data shows an identifiable feature. I tested a new algorithm written in Python that was successful at recognising this feature. I then added two graphs of processed data to my real-time display software. These can be seen as the bottom two graphs in Figure 1. These plots are used to help identify the target motion.

I ported the new algorithm to the C programming language, and added it to the gesture recognition firmware running on the T-Watch S3. When I recreated the target movement, the watch detected the motion and sent a trigger signal to the receiver unit.

Now I had a system ready for real-world testing!

DATA FROM THE T-WATCH S3

As mentioned earlier, The ESP32-S3 is the MCU used on the T-Watch S3. I use VSCode with the PlatformIO plug-in and the Arduino environment for firmware development on the MCU. **Listing 1** and **Listing 2** show the C-language code used to create and send the accelerometer data to the serial port. In Listing 1 I define a structure called `accelerometer` to store the accelerometer sensor data. The x, y, and z values of acceleration are read from the BMA423 accelerometer over I²C and stored in the `acc_x`, `acc_y`, `acc_z` elements of the accelerometer structure. An absolute value of acceleration calculated from all 3-axes can be calculated using the `calc_abs` function and stored in the `acc_abs` element of the accelerometer structure.

To write data to the serial port, I leverage the DebugLog library [9]. This library enables messages to be logged with a selectable level of importance. The different levels of importance are known as "alert levels." Different alert levels can be set for different messages. I set the alert level for logging accelerometer data to be at the DEBUG level (Listing 2).

In normal use, I don't want to stream accelerometer data to the serial port. By setting the alert level to be INFO, the DEBUG logging level is ignored as it is lower than INFO on the alert level hierarchy. I add a millisecond timestamp and a counter to the text string that contains the accelerometer data transmitted through the serial port. As we'll see later, the counter is useful to check that no data is missing. The timestamp is used to check the accelerometer sampling rate.

The function to read data from the BMA423 accelerometer is supplied by Lilygo in their Lilygolib library. This library class is instantiated in an object called `watch` in the `Lilygolib.h` file. I said that I programmed using the C programming language. However, under the Arduino framework this is not strictly true. I access C++ like classes.

Listing 3 is a highly condensed version of my `main.cpp` file, which shows the code used for handling the accelerometer data. I define a structure of type accelerometer called `watch_acc`. I plan to replace the superloop structure with a simple scheduler. The code is "good enough" for testing, but I'd like to make the architecture more modular to enable activating selected functionality using a configuration file.

To send the accelerometer data as a string to the serial port, I send the memory address where the `watch_acc` accelerometer structure is stored to the `log_acc` function using:

```
log_acc(&watch_acc);
```

So long as the logging level is at DEBUG or lower, a string of data is constructed and written to the serial port. The next question is: How do we enable the PC at the receiving end of the serial connection to use this string?

PYTHON REAL-TIME DATA DISPLAY

A Python script runs on the PC that is connected with the T-Watch S3. This script displays a graph that updates in real time, showing raw and filtered accelerometer data from the watch. The script runs across three threads so that the graph updates smoothly. I split the script across several classes, each one in a separate file.

```
#ifndef ACCELEROMETER_H
#define ACCELEROMETER_H
#include <cstdint> // needed for int16_t
typedef struct accelerometer {
  // accelerometer x-axis value
  int16_t acc_x;
  // accelerometer y-axis value
  int16_t acc_y;
  // accelerometer z-axis value
  int16_t acc_z;
  // absolute amplitude of combined x,y,z accelerometer
values
  int16_t acc_abs;
} accelerometer;
int16_t calc_acc_abs(struct accelerometer *acc);
void log_acc(struct accelerometer *acc);
#endif
```

LISTING 1
C-language code that defines a structure called accelerometer.

```
#include "accelerometer.h"
// Arduino.h needed for sqrt
#include <Arduino.h>
#include <cstdint> // needed for int16_t
// must be defined before #include <DebugLog.h>
#define DEBUGLOG_DEFAULT_LOG_LEVEL_DEBUG
#include <DebugLog.h>
// uncomment following line to disable logging
// #include "DebugLogDisable.h"
// acc_abs is the absolute value of all 3 axis
int16_t calc_acc_abs(struct accelerometer *acc){
  int16_t acc_abs;
  // sqrt is in Arduino.h
  acc_abs = sqrt(acc->acc_x * acc->acc_x + acc->acc_y * acc->acc_y + acc->acc_z *
acc->acc_z);
  return acc_abs;
}
// log accelerometer data
void log_acc(struct accelerometer *acc) {
  static uint16_t counter=0;
  unsigned long currentMillis = millis();
  LOG_DEBUG("ST", "m: ", currentMillis, "c: ", counter, "x: ", acc->acc_x, "y: ", acc-
>acc_y, "z: ", acc->acc_z, "EN", "\r");
  counter += 1;
}
```

LISTING 2
The alert level was set for logging accelerometer data to be at the DEBUG level.

The main thread is created when the `Handshake` class is instantiated in `main.py`. This thread continuously updates a graph with parsed sensor data. Instantiating this class also creates two threads called `serial_connection` and `file_thread`. Without using threads, the script would stall when, for example, new data arriving from the watch needed to be processed. This wait would halt the graph from updating, leading to a jerky display.

A diagram showing how the different classes and threads interact is shown in **Figure 4**. The `serial_connection` thread monitors the serial port to which the watch is connected. Any incoming data is checked to ensure that it is valid accelerometer sensor data. The data can become fragmented, so the `serial_connection` thread assembles the fragments into complete scans. The complete scans of data are made available to the other threads using a publisher/subscriber mechanism. This publisher/subscriber inter-thread communication is handled using the PyDispatcher library. More on this later and in **Listing 4**.

I added the facility to save the parsed data to a text file that runs in a thread called `file_thread`. This thread instantiates the `File` class. A button on the graph GUI is clicked to start the parsed accelerometer data saving to a file. The button is created in the main thread. The status of the button is shared with the `file_thread` using a queue. The queue is the second method I use for inter-thread communication. The `file_thread` receives the parsed accelerometer data from the `serial_connection` thread using the same publisher/subscriber mechanism as the main thread uses--again created using the PyDispatcher library.

The Python code for this project can be found on my GitHub site [10]. The entry point class that starts the show is called `Handshake` and can be found in the file named `main.py`.

```
#include “accelerometer.h”
#include “LilyGoLib.h”
#define CHECK_INTERVAL_MS 50 // ms interval inbetween
checking accelerometer
#define DEBUGLOG_DEFAULT_LOG_LEVEL_DEBUG
struct accelerometer watch_acc;
void setup()
{
  Serial.begin(115200);
  // remove serial timeout lag if no serial output
available
  Serial.setTxTimeoutMs(SERIAL_TIMEOUT);
  // watch defined in LilyGoLib.h
  watch.begin();
}
void loop()
{
  // send accelerometer data to serial port
  log_acc(&watch_acc);
  // check accelerometer every CHECK_INTERVAL_MS ms
  if (lastMillis < millis()) {
    // getAccelerometer is defined in SensorBMA423.hpp
as:
    // bool getAccelerometer(int16_t &x, int16_t &y,
int16_t &z)
    watch.getAccelerometer(watch_acc.acc_x, watch_acc.
acc_y, watch_acc.acc_z);
    // update the absolute acceleration from all 3
axis
    watch_acc.acc_abs = calc_acc_abs(&watch_acc);
    lastMillis = millis() + CHECK_INTERVAL_MS;
   }
   delay(CHECK_INTERVAL_MS);
}
```

LISTING 3
Snippets from `main.cpp`.

SERIAL PORT MONITORING THREAD

The Python threading library is used to create the `serial_connection_thread` that continuously monitors the serial port. This thread is created in this line in `main.py`, and collects data coming over the serial port:

```
serial_connection_thread=threading.
  Thread(target=Serial_Connect)
```

The serial connection is created in the class `serial_connection.py` in the lines:

```
try:
serial_connection = self.serial_
  connect(serial_port, baud)
except AttributeError as e:
utilities.exit_code(‘no serial
  connection found, error code:
  \n{}’.format(e))
```

This serial_connection object is then passed to the `get_bytes` method, which continuously tries to read bytes from the `serial_connection` object.

An example of a complete scan of accelerometer data that is written to the serial port by the T-Watch S3 is:

ST m: 10041 c: 57 x: 228 y: 270 z: -369 EN

I placed the “ST” and “EN” markers to mark the start and end of a scan. The “m” value is the number of milliseconds since the thread started. The “c” value is an increasing counter. The x, y, and z values are the accelerometer readings. Any bytes read by

RIGOL
Performance That's Easy To See
DM858 Series Digital Multimeter
• 5½ digit reading resolution
• Up to 125 rdgs/s measurement speed
• Up to 500,000 points logging memory
• Measures 11 input signals
• 7" color touchscreen display
DM858
DM858E

Cambrionix
CHARGE • CONNECT • MANAGE
Intelligent USB Hubs For Mass Mobile Device Management
ThunderSync3-16
Increase your productivity and ensure reliability;
charge, connect and manage all your mobile devices
using our smart USB hubs and software
Supersync15
PowerPad15S

pico
Technology
PicoScope® 2000 Series
Like a benchtop oscilloscope, only smaller and better
• 2 channel, 4 channel and MSO models
• 6 instruments in one
• Ultra-compact design
• Up to 100 MHz bandwidth
• Up to 128 MS buffer memory
• Decode over 30 serial protocols as standard
• USB connected and powered
• Windows, Linux and Mac software
From $129

QuantAsylum
Affordable and Powerful Audio Analysis
QA403 24-Bit Audio Analyzer
$590
Free Shipping
• 24-bit ADC/DAC
• Up to 192 kSa/s
• Fully isolated from PC
• 8 input gain ranges (0 to +42 dBV); 4 output gain ranges (-12 to +18 dBV).
• Max AC input = +32 dBV = 40 Vrms. Max DC input = ±40 V, Max ACPEAK + DC = ±56 V.
• USB-powered
• Built-in Attenuator

Not your average web-store...
Your source for technical advice, recommendations and novel products.
Keep checking for our newest listings!
•Saelig Company, Inc.•www.saelig.com•sales@saelig.com•1-888-7SAELIG•
(1-888-772-3544)

serial_connection are sent to a parser object. As the sensor data can become garbled or fragmented, the parser object checks for valid data and assembles complete scans of sensor data from fragments. Incomplete scans cause a "missing scan" message to be displayed. The program discounts the fragmented scan and carries on to try to read the next scan.

SENSOR DATA WITH PYQTGRAPH

The Handshake class uses a subscriber created using the PyDispatcher library [11]. The subscriber receives complete scans of accelerometer data as soon as they are published by the serial_thread. The Handshake class shown in **Listing 5** instantiates a DataFrame object called self.df. This DataFrame is updated with the new sensor data in the line:

```
self.df = self.dataframe.update_dataframe(message)
```

where message is the sensor data received using the PyDispatcher subscriber.

X, y, and z accelerometer data are extracted from the complete scans using regular expressions in the Dataframe object. The x, y, and z accelerometer values are then added to a Pandas DataFrame, along with any data created by processing these values. Pandas is a popular Python library for data processing [8]. It uses a structure called a DataFrame for storing and processing data.

A Pandas DataFrame is like an array, except that different columns can have different types. For instance, one column can be a timestamp and the next column could be an integer representing a sensor sample.

The number of lines in this DataFrame is limited. When a new row of data is added to the DataFrame, the oldest row of data is removed.

The real-time graph showing the sensor and processed graphs is created using the PyQtGraph library. This runs in the main thread by default. Any plotting objects instantiated using the PyQtGraph class must run in the main thread.

Qt is a cross-platform software development framework, and PyQtGraph is based on Qt's GraphicsView framework.

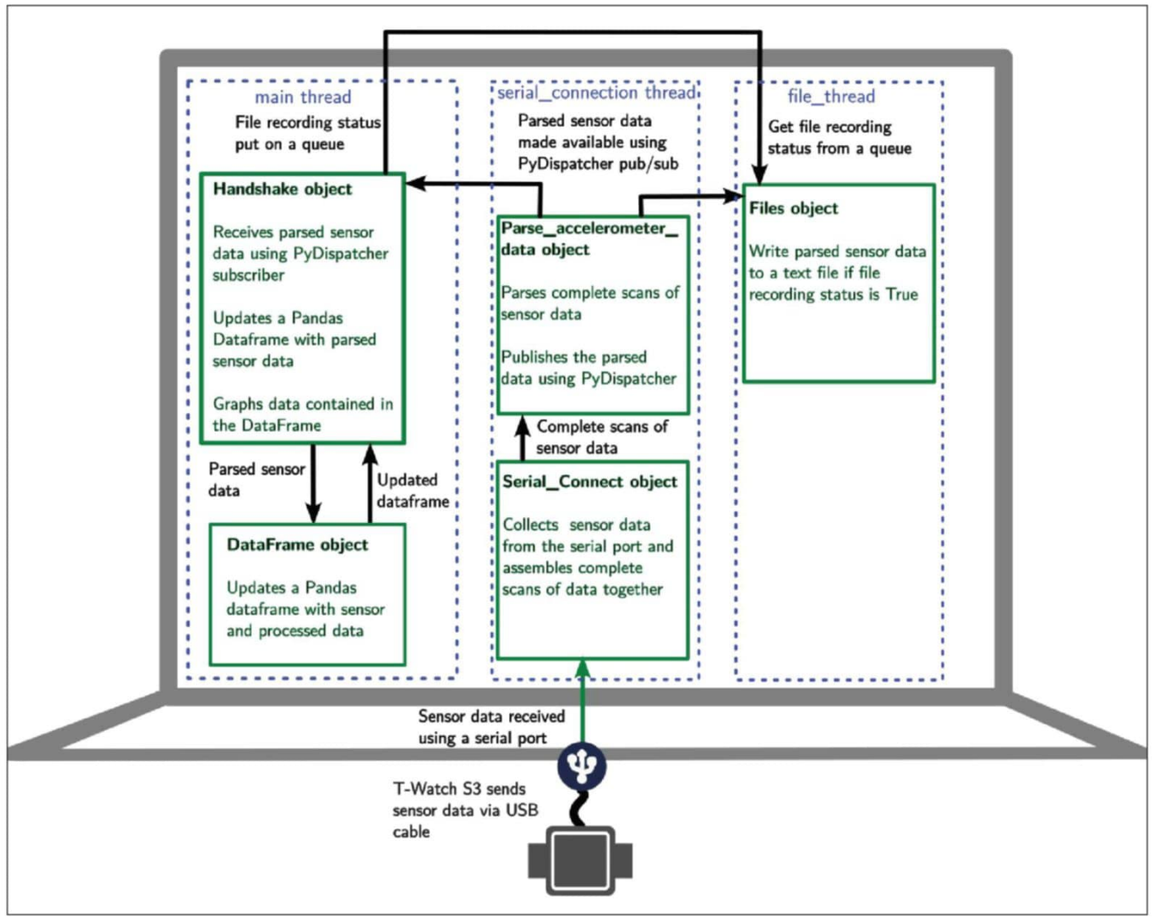

main thread
File recording status put on a queue
Handshake object
Receives parsed sensor data using PyDispatcher subscriber
Updates a Pandas Dataframe with parsed sensor data
Graphs data contained in the DataFrame
Parsed sensor data
Updated dataframe
DataFrame object
Updates a Pandas dataframe with sensor and processed data
serial_connection thread
Parsed sensor data made available using PyDispatcher pub/sub
Parse_accelerometer_data object
Parses complete scans of sensor data
Publishes the parsed data using PyDispatcher
Complete scans of sensor data
Serial_Connect object
Collects sensor data from the serial port and assembles complete scans of data together
file_thread
Get file recording status from a queue
Files object
Write parsed sensor data to a text file if file recording status is True
Sensor data received using a serial port
T-Watch S3 sends sensor data via USB cable

FIGURE 4
Project software architecture.

PyQtGraph supports two Python wrappers for this framework--PyQt and PySide. I use PySide6.

As with many graphical user interface (GUI) packages, the display is created using several classes. There are several ways we can create plots using the PyQtGraph library. However complex the following code seems, I have reduced the number of layers of widgets I use to create graphs over the last few years!

DISPLAYING X-AXIS ACCELEROMETER DATA

Figure 5 shows the arrangement of the different PyQtGraph classes used to create the display. I will explain how I create and display the x-axis accelerometer data. Please see my GitHub site [10] for code that shows how to create and display data from all three accelerometer axes and the processed data created from the x, y, and z data.

First of all, I instantiate a GraphicsLayoutWidget. According to the "readthedocs" documentation for PyQtGraph, GraphicsLayoutWidget is a "convenience class consisting of a GraphicsView with a single GraphicsLayout as its central item." Using a GraphicsLayoutWidget allows us to display several different graphs, using the same widget. By default, as we add plots, the plots are displayed underneath each other in rows. We can split each row into multiple columns and have graphs displayed beside each other, but I did not do this to keep things simple.

The GraphicsLayoutWidget is instantiated in the line:

```
self.win =
pg.GraphicsLayoutWidget(show=True,
title='T-Watch accelerometer data')
```

We now add a plot widget to the grid of plots using the addPlot() method. This line adds the plot widget, which will contain the x-axis data from the accelerometer to the GraphicsLayoutWidget:

```
self.p_xacc = self.win.addPlot(title='acc_x')
```

However, we still need to add data to the plot widget. This line adds where the data can be found to the self.p_xacc plot widget:

```
self.curve_xacc = self.p_xacc.plot(pen=acc_pen)
```

Each time that we change the data in self.curve_xacc, the plot widget p_xacc will automatically update to display the new data. If we update self.curve_xacc fast enough, the plot appears to scroll smoothly.

How we achieve this update speed is the crux of real-time data display. In PyQtGraph, we can use the QTimer class to periodically call a method that updates self.curve_xacc. As long as the time between these updates is small enough, we see a smoothly scrolling graph line.

The class that runs in the main thread and kicks everything else off is called Handshake, and is in the script called main.py. This class and the QTimer object are instantiated in these lines:

```
if __name__ == '__main__' :
  handshake = Handshake()

  timer = pg.QtCore.QTimer()
  timer.timeout.connect(handshake.timer_
timout)

  # timer units are milliseconds. timer.
start(0) to go as fast as practical.
  timer.start(UPDATE_MS) # timer timeout
in ms
  pg.exec()
```

The constant UPDATE_MS defines how often the QTimer object runs. The object runs the timer_timeout method in the Handshake class every UPDATE_MS milliseconds. The timer_timout method calls the method that updates self.curve_xacc in these lines:

FIGURE 5
The PyQtGraph classes display the x-axis accelerometer data.

ABOUT THE AUTHOR

Matthew Oppenheim works in marine geophysical survey and offshore construction. He loiters at InfoLab21, Lancaster University when not working at sea. At InfoLab21, Matthew works on assistive technology and how to deploy it in the Real World. Visit www.mattoppenheim.com for more details of these projects. Please email matt@mattoppenheim.com with any comments.

2165451

```
def dispatcher_send_data(self, data):
        ''' Publish data '''
        dispatcher.send(signal=ds.PARSER_SIGNAL, sender=ds.PARSER_SENDER, message=data)
```

LISTING 4
Creating a publisher, from `Parse_accelerometer_data.dispatcher_send_data method`.

```
def dispatcher_receive_data(self, message):
        ''' Received data from dispatcher set up in serial_connection.  '''
        # message is a acc_data structured tuple with data for a single accelerometer x,y,z scan
        self.df = self.dataframe.update_dataframe(message)
        # update sensor columns to self.df
        self.df = self.imu.update_df(self.df)
        # for debugging:
        self.log_df()
```

LISTING 5
Creating a subscriber, from Handshake.dispatcher_receive_data method.

```
acc_x = self.df['acc_x'].to_numpy()
self.curve_xacc.setData(acc_x)
```

`acc_x` is updated with the x-axis accelerometer data contained in the Pandas DataFrame. `self.curve_xacc` is then set to contain this data. The plot widget then updates and display this data. As long as we update the plot at a rate that is at least 12Hz, the graph appears to scroll smoothly.

Please note that `UPDATE_MS` only controls the period at which the graph updates. The rate at which the accelerometer is sampled is controlled by the firmware running on the T-Watch S3. Since this is likely to be higher than the rate at which the graph is updated, multiple data points are added to the graph each time the graph updates.

Data for the y-axis and z-axis are displayed by adding plot widgets to the same `GraphicsLayoutWidget` that the x-axis data is displayed on, and by assigning these widgets to contain the relevant sensor data from the Pandas DataFrame.

SAVING SENSOR DATA

The `Handshake` class creates a `file_thread` to enable parsed data to be saved to a text file in the line:

```
self.file_thread = threading.Thread(target=Files, args=(self.queue_out,)) # starts in a thread
```

The `file_thread` instantiates the class `File`. The code for this is in `file.py`. Each time a file is created, it is named with a timestamp in the filename, so that old data is not overwritten. Once the file is created, any new sensor data is written to the file. `file_thread` receives new sensor data by using a subscriber created using the PyDispatcher library. The `serial_thread` file publishes the parsed sensor data to which the `file_thread` subscribes.

The `file_thread` reads a flag that controls if the sensor data is written to file. This flag is read from a queue object. The flag is placed on the queue by the main thread and toggled True or False whenever a button on the GUI running in the main thread is clicked.

Examples of creating a publisher and a subscriber that sends and receives sensor data between threads using the PyDispatcher library are shown in Listing 4 and Listing 5. By using unique signal and sender names, we can set up a publish/subscriber messaging system between threads.

SUCCESS! ALGORITHM UPDATED

At the start of this project, I wrote C code that should have triggered on what I imagined the accelerometer data would look like for the gestures that I wanted to recognise. By graphing and processing the sensor data on a PC using Python, I radically reduced the development time of an algorithm that identifies the target hand motion.

The updated algorithm is successful at detecting the slow-rolling hand motion that I wanted to recognize. The participant with cerebral palsy is now successful at operating environmental controls and triggering recorded speech using the Handshake system. This is a first step toward the use of more complex communication software. The Handshake system is under test.

I created the algorithms using traditional signal-processing techniques, which is a testament to my reading all 100 articles of Robert Lacoste's column, "The Dark Side," in *Circuit Cellar*. I thank the team at National Star College, Gloucestershire, United Kingdom for helping with testing Handshake, and for their many ideas on how to proceed with the project.

Additional materials from the author are available at:
www.circuitcellar.com/article-materials

References [1] to [11] as marked in the article can be found there.

TECH FEATURE

CIA for SBCs (and More)

Securing Embedded Systems from Evolving Threats

By
Curtis Franklin,
Editor-In-Chief

Security for embedded systems takes many forms to protect many components from many threats. For engineers and developers, the key is keeping everything working together for the highest security with the lowest negative impact.

Cybersecurity has many qualities. Simplicity tends not to be among them. And as the processors, computers, controllers, and components used to create embedded control applications have become more capable and complex, security simplicity has receded farther and farther from view.

Embedded system security in 2025 is driven by a variety of factors. First, of course, is the rising complexity and sophistication of the threats facing embedded systems. This is tied directly to the increasing desirability of embedded control and operational technology as targets for malicious activity. Those increasingly sophisticated attacks are being directed at an attack surface that's growing by leaps and bounds.

In security, the "attack surface" is the collection of vulnerable points that present opportunities for malicious activity. Every interface, link, connection, and port presents just such an opportunity and the number of these points has grown by leaps and bounds as embedded systems have become more connected to one another, to back-end IT, and to cloud services.

The connection to systems and services beyond the embedded has brought elements from corporate governance to federal regulation into play. These elements can be prescriptive but tend to be outcome-oriented, though outcomes that are defined by the absence of something can be tricky in their own right.

So, with many factors pushing embedded system security forward, let's look at some of the specifics and how they're having an impact on the systems you build.

WALKING DOWN THE CHAIN

To understand threats to embedded systems and how best to deal with them it's helpful to have a common language to use when talking about the subject. One of the most commonly used sets of definitions comes from Lockheed-Martin in the form of the cybersecurity kill-chain [1], shown in **Figure 1**. This model begins with reconnaissance and ends with "actions on objective," whether that objective is gathering data, extorting funds through ransomware, or causing damage to operational systems. When presented visually, the action moves from left to right as the malicious campaign develops.

Another model, which goes into rather more than the seven steps of the kill-chain, is the ATT&CK matrix [2] developed by MITRE and shown in **Figure 2**. ATT&CK is a matrix that includes 245 techniques within 14 tactics. ATT&CK is comprehensive, complex, and widely used in military, critical infrastructure, and large enterprise cybersecurity. As with the kill-chain, ATT&CK begins with reconnaissance and ends with the attack's impact. As with the kill-chain, the malicious activity moves from left to right as the campaign unfolds.

That walk from left to right for the threat models leads to one of the goals of cybersecurity: "Shift-left" is the idea of detecting and disrupting a malicious campaign as early as possible in the attack process. It might seem intuitive, but attacks tend to do less damage and be easier to remediate if they are stopped before they've had a chance to move in, forward all their mail to your system, and invite the family for the holidays.

It is also easier to find and disrupt an attack if the attacker is using a single threat against a single vulnerability. Unfortunately, that is the case in fewer and fewer malicious campaigns in 2025. Sophisticated campaigns that use combinations of attacks against multiple vulnerabilities have become more common, and simple defenses that are successful against all attacks less common as both embedded systems and those who would attack them have become more sophisticated.

MORE THREATENING THREATS

The most significant vulnerabilities in today's embedded systems stem from many if the same sources as those for general IT systems: software flaws, hardware weaknesses, network exposure, and the widespread use of legacy code. Attackers are exploiting these issues across an attack surface that grows with each new cloud service, each new connected control system, and each new integration of control system and back-end analytical engine. The result is a swelling roster of severe real-world impacts.

Each of the vulnerability sources is significant, but they don't exist in individual silos. In many cases, threat actors exploit multiple vulnerabilities to craft sophisticated attacks that are more challenging to detect in their early stages and harder to mitigate after discovery. Let's look at some of the vulnerabilities frequently used in successful attacks.

Outdated and unpatched legacy code is highly vulnerable to exploits such as buffer overflows, insecure APIs, and defective access control. This is especially true when the legacy code dates from an era in which security wasn't seen as a significant concern for embedded systems. Memory safety issues remain a leading source of system compromise, and to compound the danger, legacy systems often lack the ability to be updated easily.

The difficulty in updating still sometimes extends to the default user names and passwords. Devices frequently ship with preset weak passwords or weak authentication, making brute force and credential stuffing attacks straightforward for adversaries seeking access. When the default passwords and credentials are also available through a simple web search, the vulnerability becomes critical.

Even when software and credentials can be updated, it's no guarantee that updates will come in a timely manner. Many embedded systems never receive regular firmware updates, while third-party code modules (often open source) introduce unmonitored vulnerabilities that persist in the field. Unlike desktop computer systems, where the expectation is that vulnerabilities will be actively sought out and rapidly remediated, vulnerabilities in embedded system components can be in place for years once introduced.

Network and management interfaces and services have been frequent targets for the attention of threat actors. Insecure services, such as management ports or embedded web servers, are commonly left exposed on institutional networks or even the public internet, leading to a surge in device-level exploits, remote code execution, and unauthorized system control.

Vulnerabilities that exist in hardware can as easily be the result of issues in component design as from application or operating system software development. Devices lacking secure hardware features are exposed to side-channel attacks like power analysis, electromagnetic analysis, and hardware trojans inserted via the supply chain. These techniques bypass software protections

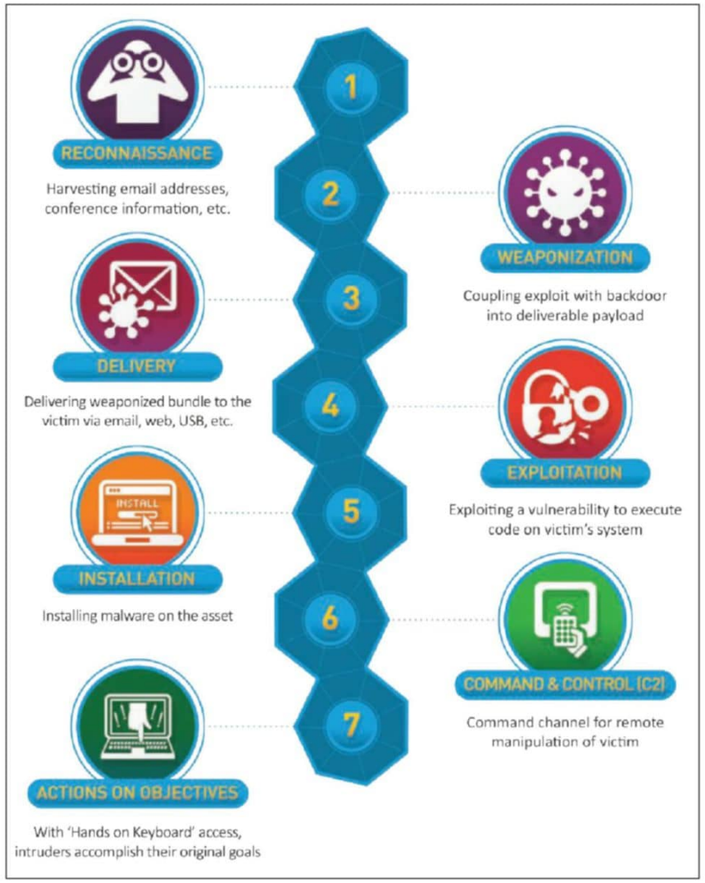


FIGURE 1
Lockheed-Martin's kill-chain is used by many in cybersecurity to describe the nature and structure of attacks.

entirely. Attackers leverage the supply chain to insert malicious code, backdoors, or counterfeit components, compromising systems even before deployment and multiplying the impact of the attack, as it affects hundreds or thousands of vendor customers rather than a single system or company.

Adversaries are leveraging AI to significantly enhance the impact of their attacks, automating scanning, vulnerability discovery, lateral movement, and credential harvesting at an exceptionally high speed. Coordinated campaigns, leveraging open-source AI tools, can identify and exploit new weaknesses before traditional defenses can respond.

Attackers further increase the impact of their campaigns when they design adversarial inputs to fool embedded AI models (object recognition, anomaly detectors, etc.), causing devices to misclassify threats or execute unsafe actions—critical in safety/security contexts like autonomous vehicles and smart cameras.

Safety and security applications are among the situations for which data integrity is critical. As part of attack campaigns, threat actors corrupt input/training data to manipulate embedded AI outputs or extract proprietary models by analyzing device responses ("model inversion")—potentially revealing sensitive logic or secrets. And that's assuming that the threat actors don't use the most effective attack vector of all: social engineering. Sophisticated phishing, impersonation, and deepfake techniques driven by generative AI increase the success rates of targeting employees or automated systems linked to embedded environments.

Modern defenses and attacks both depend on AI's ability to process data at scale and adapt rapidly—driving an escalating race between attackers and defenders in the embedded space.

BUILDING BETTER DEFENSES

Defenders haven't been idle while threat actors have improved their targeting and exploitation tactics. Technologies deployed to secure embedded systems in 2025 combine hardware-based safeguards, advanced cryptographic solutions, secure software practices, and real-time monitoring. It is important to understand that the list of defensive strategies isn't a "choose one" situation in which you look for the cybersecurity silver bullet that will keep all threats at bay. Instead, these measures work together to create a layered defense against modern threats.

Starting at the beginning of the application's uptime, devices use secure boot processes and hardware root of trust (e.g., TPM, Arm TrustZone) to ensure that only authenticated code is executed and cryptographic keys remain protected. These strategies are often combined with memory security, in which memory is secured with cryptographic protections, anti-rollback counters, power-loss protection, and error correction. Secure boot, hardware root of trust, and memory security are now standard in mission-critical devices to prevent firmware tampering and side-channel attacks.

BEYOND THE BOOT

Cybersecurity doesn't end when the boot process completes. Because there are vulnerabilities that come into play once the application code begins to execute, there are many defensive tools and techniques that begin their protective activities at the same time.

Open-source advanced cryptographic libraries like wolfSSL provide FIPS-validated encryption, TLS, and even post-quantum cryptography to embedded projects. The challenge is convincing software developers that the advantages of the libraries is worth the cost of learning to include them in the first place. The same challenges exist when the encryption of data in transit (TLS/DTLS, VPN) and at rest (on-device AES/NAND encryption) are deployed broadly across embedded device classes. Showing developers that greater security is worth both their time to implement "secure by design" [3] and the impact on system performance that any additional code will inevitably incur is frequently the most difficult part of improving embedded system security.

SECURE SOFTWARE DEVELOPMENT

While software design and programming techniques are critical for building secure systems, other tools can help enhance the foundational components that boost security. Memory-safe programming languages, such as Rust and modern C++, are being adopted by a growing number of embedded software developers, helping to prevent memory safety bugs by design, especially in new codebases. And this more secure software can maintain its enhanced security with updates that apply more easily.

FIGURE 2
The MITRE ATT&CK Matrix is a comprehensive model for understanding the processes of different cybersecurity attacks.

Many systems are being deployed with a variety of secure update mechanisms. Devices are being configured to feature secure over-the-air (OTA) updates with digital signature verification, rollback prevention, and atomic swaps (peer-to-peer exchanges of software assets across separate blockchain networks), ensuring that only authorized firmware runs and that updates can be safely reversed in the event of a compromise. The combination of software designed to be secure and deployment methods intended to keep the software secure is being wrapped up in overall software disciplines that support the goal of software security

DevSecOps, in which software development, security, and operations are integrated to form a cohesive whole, is common in enterprise IT but has lagged in its adoption among OT practitioners. That is beginning to change as security-by-design approaches, continuous integration of security testing, Software Bill of Materials (SBOM) generation, and compliance with NIST and ISO standards are now common practices in large-scale embedded system deployment and are gaining acceptance in smaller operations.

The increasing shift to more powerful processors for embedded control means that certain levels of more complex security techniques are becoming possible. Address space layout randomization (ASLR) and stack protection are now being implemented even in real-time operating system (RTOS) environments to counter memory-based exploits. The greater processor capabilities also bring the opportunity for more aggressive real-time monitoring of individual components and the system as a whole to detect anomalies earlier in their operation.

Intrusion Detection and Prevention Systems (IDS/IPS) for embedded systems, for example, monitor both network and internal process activity to identify and mitigate threats in real time. For those who are fuzzy on the difference between IDS and IPS, an IDS detects intrusions and passes data to other systems that can act to disrupt and prevent the activity. IPS systems have the capability to both detect and prevent the intrusion, though in most cases, they also pass data to other systems for analysis or network protection modification to help shift defense left.

AI and machine learning (ML) are also contributing to early detection of traffic and behavioral anomalies. The two technologies are applied to real-time network monitoring and to log files for training, enabling the detection of unusual behavior earlier than human analysts would typically recognize a problem. When combined with kernel and process isolation, these tools can help prevent lateral movement and privilege escalation.

These combined strategies reflect a shift toward proactive, multi-layered defense—making attacks significantly harder and reducing the long-term risk for embedded systems. The layers continue with hardware components designed to make successful intrusions even less likely to come through an embedded system.

SECURE HARDWARE PLATFORMS

Hardware-based security in embedded systems safeguards devices by integrating protective functions directly onto the silicon, using dedicated components to resist both conventional and physical attacks. This approach surpasses what software-only defenses can provide, establishing a robust foundation for the security of modern embedded systems.

Trusted Platform Module (TPM) chips are cryptographic processors that generate, manage, and store cryptographic keys, certificates, and measurements of critical boot processes in a secure manner. TPM-based secure boot checks the integrity and authenticity of firmware at each stage, preventing unauthorized or malicious code execution. TPMs also enable device identity, disk encryption (e.g., LUKS), remote attestation, and protection of communication keys—all isolated from standard system memory and software access.

Just as software-based secure boot processes can prevent unauthorized software from loading before memory-based security software initiates, hardware-enforced secure boot ensures that only signed and authorized firmware and operating systems are loaded onto the device. Verification occurs at each step, starting from the lowest-level loader and progressing up to the application layer, utilizing hardware root-of-trust chips such as TPM, Secure Elements, or embedded specialty microcontrollers.

Once the system has booted and the application software begins to load and operate, technologies such as Arm TrustZone and Intel SGX create isolated areas for running sensitive code and storing data, which are inaccessible to normal processes and malware. Trusted Execution Environments (TEEs) protect against privilege escalation, runtime exploitation, and the leakage of secrets.

Secure memory chips offer encrypted storage, anti-rollback mechanisms, and tamper detection to safeguard secrets and data at rest. Hardware secure elements (e.g., NXP A700x, Infineon Optiga) are used for storing root keys, enabling secure authentication, and supporting post-quantum cryptography for edge devices. The stored keys and secrets can best be generated by hardware-based cryptography to accelerate and protect cryptographic operations

(AES, RSA, ECC), leaving sensitive keys and computations outside the reach of software-level attacks or memory dumps.

Hardware integration is important because it provides defensive layers that resist both remote and physical attacks, including device probing, reverse engineering, and supply chain threats. Security mechanisms like secure boot and encryption can be enforced regardless of software vulnerabilities, making it significantly harder to compromise the device even if the operating system is breached.

Hardware also enables tamper detection and zeroization (instant erasure) of secrets upon sensing physical intrusion attempts. Leading hardware-based security solutions build a trustworthy root, provide encrypted data and identities, and power real-time verification of device integrity—establishing strong barriers against today's advanced embedded threats.

FIGURE 3
NVIDIA Jetson AGX Orin Modules have sufficient processing power to bring AI to embedded systems at the network's edge.

A SECURE SYSTEM SAMPLE

Many embedded products today integrate hardware security components such as TPMs, secure elements, and TrustZone-enabled microcontrollers. Here are examples featuring these technologies across MCU, SoC, and module levels.

Microchip PIC32CM LS60 MCU—This MCU integrates a robust, secure subsystem (Microchip Trust Platform) and Arm TrustZone in one MCU for easy development of secure IoT, medical, and consumer products.

Infineon OPTIGA Trust Secure Elements—These elements provide tamper-resistant chips for secure boot, device authentication, and TLS communication. The Infineon OPTIGA Trust Secure Elements are widely used in IoT sensors, automotive subsystems, and smart meters.

NVIDIA Jetson AGX Orin Modules—High-end edge AI platforms, shown in **Figure 3**, that incorporate HSMs and secure boot features for robotics, video analytics, and industrial automation.

NXP EdgeLock Secure Element—Protects keys, identities, and firmware in gateways, industrial automation controllers, and embedded edge nodes.

NXP LPC5500 Series MCU—Features Arm Cortex-M33 with TrustZone for hardware-enforced partitioning and crypto accelerators for secure IoT and sensor designs.

Renesas RA8T2 and RA4C1 MCUs—Utilize Arm Cortex-M85/M33 cores with TrustZone for motor control and industrial/consumer systems, offering hardware separation and encryption.

Qualcomm Robotics RB5 Platform—Combines AI acceleration with hardware cryptography and security features for robotics, drones, and edge AI devices.

Rockchip RK3588 SoC—Used in single-board computers and edge devices, the RK3588 (shown in **Figure 4**) combines CPU, GPU, and secure NPU with hardware isolation and secure boot for embedded AI and gateways.

STMicroelectronics STSafe Secure Element—Provides crypto key storage and device identity protection for smart objects, payment terminals, and wearables.

These products are examples of the widespread adoption of hardware security. As pieces of the layered approach to embedded system security, they help provide robust protection for device identity, firmware integrity, and data confidentiality across diverse embedded applications.

INTELLIGENT SECURITY

AI is playing a transformative role in embedded system security for both defenders and attackers, bringing new capabilities—and new risks—to the field. Security components with AI onboard are increasingly used for both rapid response to active threats and predictive actions for maintenance and intrusion defense.

AI-driven autonomous defense systems in embedded cybersecurity represent a major shift toward proactive, self-adaptive, and rapid protection mechanisms. These systems utilize machine learning and agent-based automation to identify, respond to, and even remediate threats without human intervention—often in real-time.

For intrusion and anomaly detection, lightweight ML models, including compressed neural networks, are increasingly deployed for real-time intrusion detection on resource-constrained embedded and IoT devices. These systems can analyze system call sequences, network traffic, and behavior logs to spot suspicious activities and novel attacks. Once identified, AI agents can automatically respond to detected threats, isolate compromised components, and trigger secure updates with minimal to no human intervention, making them an ideal set of technologies for embedded systems.

AI agents embedded in security stacks continuously monitor system and network behavior, searching for anomalies, abnormal device actions, or patterns of known and unknown attacks. When a threat is detected, these agents can autonomously isolate affected systems, block malicious processes, and trigger secure remediation steps.

Embedded systems also benefit from improved predictive maintenance and health monitoring. Embedded AI can forecast failure modes or detect tampering and attacks in critical systems (e.g., aerospace, defense, and critical infrastructure) before they cause real-world consequences. As part of its maintenance and health monitoring functions, edge AI enables embedded devices to conduct local, real-time screening for malware, resource manipulation, and data exfiltration—even when the embedded system is disconnected from a central server.

Unlike static rule-based systems, AI models update their detection profiles as new tactics emerge, learning from recent attacks and evolving environmental changes. This makes autonomous defense resilient against zero-days and sophisticated adversaries.

With constant monitoring available, modern embedded security solutions use AI to understand normal operation baselines and raise alarms when small deviations—potentially caused by stealthy zero-day exploits—occur.

With increased autonomy, AI defense agents themselves become high-value targets—if subverted, they can be used for large-scale, automated attacks. Building explainability, robust oversight, and strong access controls into AI-driven security systems is now considered essential for trustworthy deployment.

AI-powered autonomous defense will define the next generation of embedded security, driving ultra-fast, adaptive, and consistent protection across critical infrastructure, vehicles, and industrial edge systems.

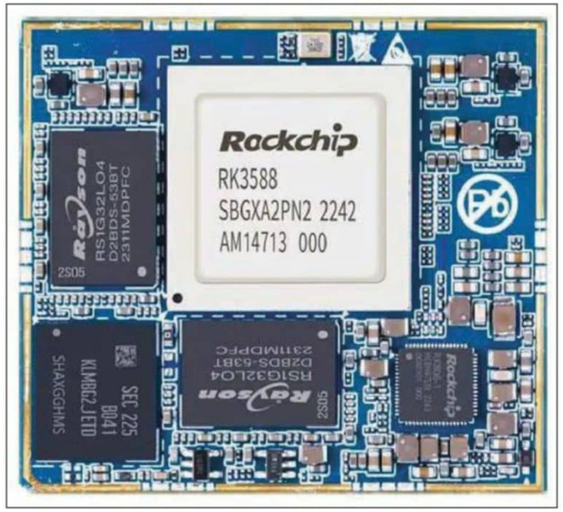


FIGURE 4
The Rockchip RK3588 SoC is an example of a multi-functional, multi-part system on a single chip that can be integrated into embedded systems where size is a significant constraint.

MOVING FORWARD

Advancements in autonomous embedded cybersecurity systems over the next five years are expected to radically improve defense capabilities, resilience, and threat anticipation, powered by breakthroughs in AI, hardware, and design philosophy.

Autonomous systems will deploy advanced machine learning models to forecast and detect threats before they occur, leveraging historical and real-time data across global device fleets. AI agents will proactively patch vulnerabilities and optimize defenses, shrinking the window between exploit discovery and mitigation.

Once intrusions are detected, AI will power dynamic, context-aware Zero Trust frameworks that continuously verify device integrity, user behavior, and environmental context before granting access, increasingly making static policies obsolete. With that, next-gen autonomous systems will feature "self-healing" capabilities: rapidly isolating, repairing, and restoring compromised components without human intervention, minimizing downtime and damage.

Security response will not involve humans and will involve central cloud services less and less as time goes on. Security intelligence will migrate from central clouds to embedded edge devices, enabling real-time threat detection/response with ultra-low latency and improved privacy. New edge AI platforms will run complex defense algorithms on embedded hardware even during connectivity loss.

Defenders will be increasingly empowered by generative AI copilots that streamline investigation, automate threat modeling, and simulate attacks to discover new vulnerabilities, keeping defensive strategies ahead of rapidly morphing AI-powered malware. And the defenses will morph and adapt as rapidly as the attacks. Systems will rely less on static signatures, instead adopting AI-driven behavioral analytics that adapt to new threats and user patterns, detecting zero-days, novel malware, and insider risks instantly.

These innovations will enable embedded systems to autonomously defend against rapidly evolving attacks and orchestrate collective responses across device swarms—setting new standards for resilience, privacy, and operational continuity in an increasingly hostile threat environment.

Additional materials from the author are available at: www.circuitcellar.com/article-materials

References [1] to [3] as marked in the article can be found there.

RESOURCES

Infineon | https://www.infineon.com

NVIDIA | https://www.nvidia.com

NXP | https://www.nxp.com

Qualcomm | https://www.qualcomm.com

Renesas | https://www.renesas.com

STMicroelectronics | https://www.st.com

TECH FEATURE

DATASHEET

Datasheet: Making the Right Voltage

DC-to-DC Converters Bring the Right Voltage to Circuits

By
Curtis Franklin,
Editor-In-Chief

The proper voltage for a circuit's needs doesn't just happen. From a single source, multiple voltages may be required to bring power to the whole system. DC-to-DC converters can turn one voltage into another, whether higher or lower, and ensure that the results are safe, reliable, and free from unnecessary electronic drama.

It would be so nice if every part of every circuit used the same voltage for operation. 5V, 9V, 12V, 27.8V—it wouldn't really matter as long it was the same from one end of the project to the other. But, of course, life isn't nearly that simple. Instead, circuits require a wide range of voltages, often with different tolerances for drift and error. Getting from the single input DC voltage to the plethora of necessary circuit DC voltages requires a DC-to-DC converter.

MOVING THE VOLTAGE

DC-to-DC converters are used whenever different sub-circuits within a device require different voltages. They can step up (increase) or step down (decrease) voltage levels, improving energy management and supporting battery-powered and embedded devices by maximizing efficiency and battery life while minimizing the number of different voltage sources required.

DC-to-DC converters can also do more than simply shift voltages; some designs provide electrical isolation between input and output using transformers, which helps protect sensitive circuits from noise and surges.

CONVERTER VARIETIES

DC-to-DC converters are often categorized into major types based on their topology and use case.

- Buck converter: This converter steps down the voltage (output voltage is lower than the input).
- Boost converter: This converter steps up the voltage (the output is higher than the input).
- Buck-boost converter: A buck-boost converter can both step up and down, and may invert the output polarity.
- SEPIC converter: The single-ended primary-inductor converter (SEPIC) can step voltage up or down with output that is not inverted.
- Isolated converter: This converter uses a transformer for input/output isolation, preventing voltage surges, drops, and spikes on the input side of the converter from reaching and damaging components on the output side.

PERFORMANCE CHARACTERISTICS

While all DC-to-DC converters perform similar core actions, there can be significant differences in the performance characteristics as they do their jobs. There are important performance characteristics to consider when selecting a DC-to-DC converter for a design.

Those characteristics may include input voltage and output voltage range; output current capability; efficiency (usually between 75% and 95%, with higher efficiency generally related to high-quality units); voltage regulation precision (often less than 1% deviation); and thermal protection and packaging style.

While it's easy to say that every DC-to-DC converter should be spec'd with the highest possible quality and performance ratings, cost and physical packaging considerations mean that it's more important to specify the right qualities than the maximum qualities. It's up to the engineering team to understand—and specify—the right qualities for the job.

NEXT MONTH'S TOPIC: Tiny Embedded Boards Send related product announcements to editor@circuitcellar.com

High-Power Managed Temps

Advanced Energy's Artesyn NDQ1300 series quarter-brick is a non-isolated DC-DC converter that provides a single, regulated, low-noise output with a digitally-controlled PMBus interface module function. It delivers a 12VDC output voltage at up to 1300W. The 97.7% peak efficiency, excellent thermal management, and operational ambient temperature range of -40 °C to +85 °C make it a good choice for use in datacom and telecommunication applications that can employ a non-isolated power conversion function and 40VDC to 60VDC input to a 12.15V output.

- 1300W continuous power
- Ultra high efficiency: 97.7% peak
- 40VDC to 60VDC telecomm input range
- Base plate for contact cooling or heatsink mounting
- Fixed switching frequency
- Pre-bias start-up capability
- Parallel operation, droop current sharing/active current sharing option
- PMBus function
- Remote control function (negative logic)
- RoHS 3.0
- Input under-voltage protection
- Input over-voltage protection
- Output over-voltage protection
- Output over-current protection
- Over-temperature protection
- Designed to meet IEC62368
- CE
- UL94 V-0 materials

Advanced Energy
advancedenergy.com

Vehicle Conversion

The 750-Watt Calex 48S12.750BCE bi-directional, non-isolated, DC-to-DC converter provides a complete solution for in-vehicle power distribution with 12V/48V battery configurations for a variety of applications, including micro and mild hybrid automotive systems.

The bi-directional DC-to-DC converter charges a low side (12V) battery during normal operation (Buck mode) and charges or assists the high voltage (48V) battery in emergency situations (Boost mode). The bi-directional DC-to-DC converter operates as an ideal current source with variable direction, allowing energy transfer between two voltage domains.

Voltage feedback maintains the output voltage within the acceptable operating range. It regulates the average current flowing between the high voltage and low voltage ports in the direction selected via CAN interface.

- Buck and Boost modes of operation
- Low Side (LS): 12V Input Voltage Range: 9V to 16V
- Low Side (LS): 12V Output Voltage Range: 6V to 16V
- High Side (HS): 48V Input Voltage Range: 32V to 63.2V
- Reverse polarity protection
- Constant Voltage and Constant Current Mode
- Internal Temperature Monitoring
- High-power density
- Efficiency up to 96.7%
- Dimensions 4.84" x 6.97" x 1.75" (123 x 177 x 44.5 mm)
- Weight 3.04 lbs. (1.38 Kg)
- CAN 2.0b Interface

Calex
calex.com

Distributed Conversion

The Cosel DCS1400B is a flexible, wide-range DC-to-DC converter that is ideal for distributed power systems. Intended for mounting within a system cabinet, the DCS1400B has good temperature dissipation characteristics and voltage isolation capabilities that make it safe to include in a wide variety of solutions, including applications in the industrial, medical, and renewable energy sectors.1

Cosel DCS DC-DC Converters offer a 1200W to 1404W output power range, up to 95% efficiency, and dimensions of 116.8mm x 12.7mm x 61mm. These converters provide constant current regulation, parallel operation, an optional built-in O-ring MOSFET, and various built-in protections, including overcurrent, overvoltage, and thermal protection. Flexible voltage settings are enabled due to the wide output voltage range, and the output voltage can be varied to near 0V.

- Wide output voltage range allows for flexible voltage settings
- Output voltage can be varied to near 0V
- Built-in overcurrent, overvoltage and thermal protection circuits
- Built-in remote ON/OFF
- Built-in Power good
- Constant current regulation
- Parallel operation possible
- N+1 Parallel Redundancy Operation possible
- Built-in ORing MOSFET (DCS1400B-P2)

Cosel
coselusa.com

DATASHEET URLS:

Advanced Energy: https://www.advancedenergy.com/getmedia/5d317ace-4d34-435e-8a5d-bcd918d1fef8/en-datasheet-NDQ1300-3-25.pdf

Calex: https://www.relec.co.uk/wp-content/uploads/2023/06/48S12.750BCE_A01.D10.pdf

Cosel: https://datasheet.octopart.com/DCS1400B12-Cosel-datasheet-188804378.pdf

DATASHEET

Datasheet: Making the Right Voltage

Conversion in the Air

Curtiss-Wright`s LV-HV-DCC DC-to-DC converter system combines a modular design with power management technology. This technology allows the units to generate higher voltages while meeting high safety standards.

These modular, scalable DC-to-DC converters are designed for deployed applications that require configurable voltage levels of up to 750V. The converters can be easily modified to meet specific customer needs due to their modular design. Each DC-to-DC converter has high-speed and low-latency RS-422/RS-485, CAN bus, and Gigabit Ethernet interfaces that enable seamless integration into different architectures and environments.

To accelerate system development and integration, Curtiss-Wright products ship with a GUI-based service application.

- Input voltage (MIL-STD-1275): 17 to 32VDC
- Output voltage: Input voltage to 750VDC
- Output power: 1 to 12kW (higher values on request)
- Operating temperature (MIL-STD-810): -40°C to 62°C
- Environmental: MIL-STD-810
- MTBF (MIL-HDBK-217 FN2, NPRD-2016): ~ 25,000 hours
- Galvanic isolated power stage
- Insulation fault monitoring guard
- Voltage and temperature protection
- Input current measurement Inrush current limitation
- Transient voltage and overvoltage protection

Curtiss-Wright
curtisswright.com

Compact Surface-Mount

HALO's family of DC11 1W and DC21 2W isolated DC-to-DC converters provides a compact surface mount voltage conversion solution with high efficiency and low ripple/noise.

Both 1W and 2W versions are available in industry-standard SMD packages and the 1W solution is available in either 1.5KV or 3KV isolation. All HALO DC converters offer continuous short-circuit protection and meet UL/EN62368-1.

Both 1W and 2W converters are available in a variety of both input and output voltages, representing both buck and boost capabilities. In both cases, the range of input and output voltage is between 3V and 24V, depending on the specific model and configuration of the particular DC-to-DC converter.

HALO's DC-to-DC converters are recognized for their overall electrical efficiency in deployment. The efficiency of the converters varies with the input and output, ranging from 77% to 90%.

- DC/DC 1W and 2W Unregulated Converters
- 1.5KVDC Isolation Voltage
- Functional Insulation
- Operating Temperature -40°C to +105°C
- High Efficiency & Low Ripple/Noise
- Continuous Short-Circuit Protection
- UL/EN62368 Recognized

HALO
haloelectronics.com

Space-Grade Conversion

The Microchip SA15-28 is an isolated DC-to-DC converter capable of delivering up to 15W of output power in a small-sized design. The SA family is space-rated and provides a radiation-hardened option with top-class TID and SEE performance for space and military applications.

With a flyback converter topology, the SA15-28 is optimized for applications that require isolated DC voltage conversion. The discrete surface-mount design facilitates customization with reasonable lead time and modest NRE cost. SA15-28 series excels in its robustness in applications with 39.6 x 10^6 hours of MTBF.

The SA15-28 is available in a 2.27″ × 1.72″ × 0.484″ max package. The SA15-28 triple output supplies three isolated, semi-regulated outputs, and is designed to be a frontend for post regulators with outputs that can be set up in series or parallel configurations.

- Up to 15W output power
- 22VDC to 40VDC input range
- Triple Isolated 5V outputs
- <1% output ripple
- Flyback topology
- Inhibit pin for electrical ON/OFF
- Low mass of 55g
- Flight proven technology with >39.6 × 10^6 hours of MTBF
- Product is classified as EAR99
- Customized input/output voltages available
- Radiation Performance:
- TID >100 krad (Si) and 30 krad (Si) ELDRS (<10 mrad/s) per MIL-STD-883 Method 1019

Microchip
microchip.com

MOSFET Buck

The Rohm BD9E104FJ is a synchronous buck DC-to-DC converter with built-in low on-resistance power MOSFETs. The converter features high efficiency at light load with a Simple Light Load Mode (SLLM).

Because of its high light-load efficiency, the BD9E104FJ is most suitable for use in equipment to reduce the standby power required. It is a current-mode control DC/DC converter and features high-speed transient response. Phase compensation can also be set easily.

The Rohm BD9E104FJ is suitable candidate for a variety of applications, including consumer applications such as home appliances, secondary power supply and adapter equipment, and telecommunication devices.

- Input Voltage Range: 7.0V to 26.0V
- Output Voltage Range: 1.0V to VIN x 0.5V
- Output Current: 1.0A (Max)
- Switching Frequency: 570kHz (Typ)
- High Side MOSFET ON-Resistance: 250mΩ (Typ)
- Low Side MOSFET ON-Resistance: 200mΩ (Typ)
- Shutdown Current: 0μA (Typ)
- SLLM Control (Simple Light Load Mode)
- Single Synchronous Buck DC/DC converter
- Over Current Protection
- Short Circuit Protection
- Thermal Shutdown Protection
- Under Voltage Lockout Protection
- Internal Soft Start
- SOP-J8 Package

Rohm
rohm.com

Regulation without Isolation

The Vicor DCM3717 is a non-isolated, regulated DC-DC converter module that operates from a semi-regulated 40V – 60V input to generate a regulated output of 10.0V – 12.5V. The DCM3717 in the SM-ChiP package configuration utilizes the Vicor-patented zero-voltage switching (ZVS) buck-boost regulator stage, followed by the Sine Amplitude Converter (SAC). The DCM offers flexible thermal management options with very low top- and bottom-side thermal impedances. SM-ChiP-based power components allow engineers to build low-cost power system solutions with small system size and weight, and large efficiency attributes quickly.

The DCM3717's collection of attributes makes it suitable for deployment in applications for data centers, high-performance computing systems (HPC), mild-hybrid and autonomous vehicles, as well as a wide range of industrial systems.

- Wide input range 40VDC – 60VDC
- Trimmable output range 10.0VDC – 12.5VDC
- 96.5% peak efficiency
- Up to 750W continuous operation
- Up to 900W or 74A transient peak
- >1MHz switching frequency
- PMBus compatible telemetry
- Internal voltage, current, and temperature shut down
- Array configuration of up to four units

Vicor
vicorpower.com

Conversion in Space

The VPT SGRBX Series is a box solution that contains four SGRB Series DC-DC converters. The individual SGRB slices are configured within the box framework to deliver up to 1600W of output power. System designers may select different SGRB Series units to be paralleled inside of the box solution to reach desired output voltages and power levels required for specific applications.

The SGRBX Series is a solution for space-rated DC-DC power conversion. VPT designed this box solution and the individual SGRB converters to thrive in the most demanding commercial, scientific, and military space environments. With GaN technology, its radiation-hardened design ensures long-term reliability.

The SGRBX utilizes SGRB converter slices with fixed-frequency, reduced-voltage switching topology, resulting in extremely low input and output noise. This solution is suitable for a wide range of high-power space applications.

- Up to 1600W output power
- Temperature range of -35 °C to +70 °C
- Undervoltage lockout
- Integrated EMI filter
- High efficiency up to 96% using GaN technology
- 80V transient with reduced output power
- Overvoltage protection
- Current limit/short circuit protection
- Power supply telemetry
- Parallel operation for higher power requirements using current share.

VPT
vptpower.com

DATASHEET URLS:

Curtiss-Wright: https://www.curtisswrightds.com/resources/lv-hv-dcc-dcdc-converter-data-sheet

HALO: https://www.mouser.com/pdfDocs/dc11-converter-datasheet.pdf

Microchip: https://ww1.microchip.com/downloads/aemDocuments/documents/IPS/ProductDocuments/DataSheets/Radiation-Hardened-Isolated-DC-to-DC-Converter-DS00005965.pdf

DATASHEET URLS:

Rohm: https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/switching_regulator/bd9e104fj-e.pdf

Vicor: https://www.vicorpower.com/documents/datasheets/ds-DCM3735S60D13L0TN1-VICOR.pdf

VPT: https://www.vptpower.com/products/datasheet/sgrbx-series-pdf

PICKING UP MIXED SIGNALS

A GPS-Disciplined Frequency Counter using the PSoC 5LP

In this month's column, Brian tasks one of his favorite mixed-signal MCUs, the Infineon PSoC 5LP, to build a frequency/period counter using a GPS satellite module to discipline the project's time base.

By
Brian Millier

Frequency counters and signal generators generally derive their accuracy from whatever accurate time base they have on-board. This is often a crystal oscillator, which may take the form of a Temperature Compensated Crystal Oscillator (TCXO) if better accuracy is needed. Oven Controlled Crystal Oscillators (OCXOs), however, are incorporated in the "Cadillac" of time-bases (ignoring very high-tech solutions like a rubidium-based frequency standard), and are much more expensive than TCXOs.

While most people are quite familiar with the location information that various GPS satellite constellations provide, everything about GPS boils down to being able to measure time extremely accurately. GPS location services work by comparing timing signals received from several different GPS satellites (the more the better) and calculating the GPS receiver's location from these time relationships. The GPS satellites themselves have extremely accurate time-bases on-board and know their own positions accurately, allowing this scheme to work. This GPS timing signal can be used to implement an accurate time-base.

The cost of GPS receiver modules has dropped substantially over the last decade or so. Most of this can be attributed to the fact that tiny GPS receivers are built into virtually every cell phone currently sold. Although this drop in price made GPS modules readily available for electronics enthusiasts and makers, they still weren't particularly useful in their earlier iterations because the receiver chip in them was not sensitive/noise-free enough to allow them to work indoors. I bought a modestly priced GPS module a few years ago, hoping it would work indoors—at least near a window, anyway. It didn't—and is buried in my junk box somewhere—since I didn't have any outdoor use for it in mind.

Today, however, even very low-cost GPS modules will work indoors—at least in the wood-frame house I live in. The GPS module that I settled on is the Grove-GPS (Air530) by Seeed Technology. [1] These modules cost about $13 and include an external antenna, which is handy if you are mounting them in a metal enclosure. **Figure 1** is a photo of this module. The "Grove" part of the model number refers to the fact that it contains a standard Grove connector, which provides power to the module and also contains the serial data lines (Rx and Tx).

The GPS signal that can be used to adjust (discipline) your time base is the 1PPS signal. This provides a 100ms pulse at a precise 1Hz rate. This 1PPS signal is available on one of the 5 pin headers found on this GPS module (but not available on the 4-pin Grove connector).

The board containing the Air530 GPS module has two LEDs. Immediately upon powering it up, a blue LED will flash every second. This LED merely monitors the Tx line, which sends out an NMEA GPS message every second. These messages are sent whether any GPS satellite signals can be found or not. It is only after the module has found at least one GPS signal that it will start outputting the 1PPS signal. There is a green LED on the module that will flash in time with this 1 PPS signal.

In practice, I found this GPS module was able to receive a sufficient GPS satellite signal when operating indoors, allowing it to provide the 1 PPS signal. It didn't have to be near a window. The Air530 contains a coin cell holder on the back side. I fitted a coin cell because this backup power source allows the module to store many parameters in the unit's RAM memory. This will greatly reduce the time it takes to achieve a GPS "lock," called a "warm start." Without the backup cell, on power up it executes a "cold start," which can take many minutes.

While I was developing the project, the GPS module (and the rest of the circuitry) was sitting on my desk and connected to my PC. I have three flat screen monitors connected to my PC, located very close to the GPS module. Under those conditions, a "cold start" sometimes failed (or I just lost patience). However, I found that the GPS module would get a GPS lock under the same conditions when the backup cell was mounted.

My first iteration of this project used the ubiquitous ESP32 MCU and a Silicon Labs Si5351 synthesizer chip. The Si5351 can generate frequencies such as the 10.000MHz time base that I intended to use for the project. It has a resolution in fractional Hz and can easily be calibrated by setting the value of an internal calibration register. My initial design used 4 TTL logic "glue" chips in addition to the ESP32 and Si5351 synthesizer. It wasn't until I got near the end of the construction/programming of the project that I encountered a fly in the ointment. It turns out that the internal oscillators used in the Si5351 operate in the 600-900MHz range. RF leakage of harmonics from the Si5351 was enough to swamp the tiny GPS satellite signals (which are in the 1176-1575MHz range). This could be remedied by placing the Air530 GPS module several feet away from the rest of the project's circuitry (containing the Si5351). However, that didn't seem practical to me.

My second attempt at this project was a complete re-design. I decided to use a 10.0MHz Voltage Controlled Crystal Oscillator (VCXO) as the time base. A VCXO has several advantages for this purpose:

- It produces no harmonics anywhere near the GPS signal frequencies.
- It combines the accuracy of a good crystal oscillator with the ability to be fine-tuned ("pulled") to the exact frequency, under voltage control.
- It is more expensive than the Si5351 Synthesizer chip, but still reasonably priced.

While the ESP32 MCU that I used in my first iteration has a very good variety of timer/counters and clocks available, I still ended up needing four TTL glue chips to complete the design. This time around, I decided to use Infineon's PSoC 5LP system on a chip. This chip family was designed by Cypress Semiconductor before they were bought by Infineon. This part contains many analog and digital function blocks, all of which are highly configurable in terms of digital bit-depth as well as user-definable digital and analog interconnection paths within the device itself [2].

I'm quite fond of the various Cypress PSoC chips and have used them in past projects that I've published in Circuit Cellar, such as the Picking Up Mixed Signals articles: "Variable

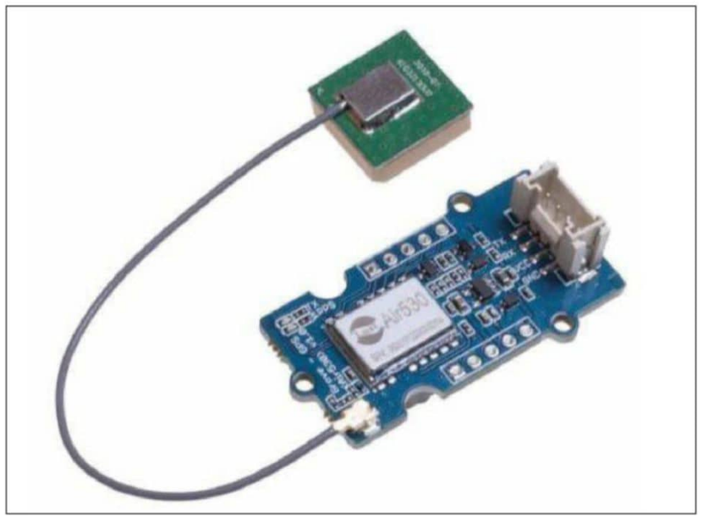


FIGURE 1
This is the GPS unit that I used for this project. It is sold by Seeed Technologies. The external antenna is useful when the GPS module itself must be mounted in a metal enclosure.

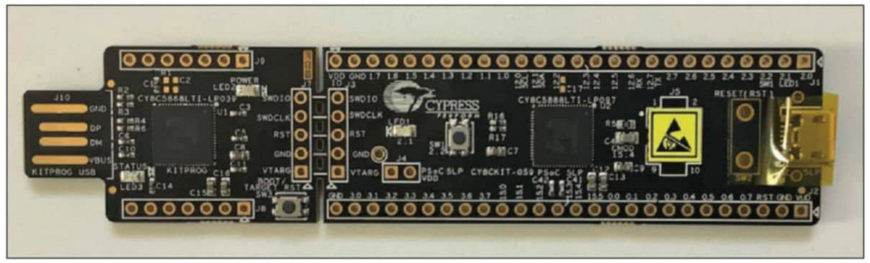


FIGURE 2
This is the CY8CKIT-059 development board for the PSoC5LP system on a chip. To the left is the CMSIS-DAP flash programmer/debugger, which can easily be snapped off when development is finished.

Frequency Drive (Part 1): Washing Machine Repurposed" [3], "Variable Frequency Drive (Part 2): Bandsaw Upgrade" [4], "Bluetooth-Enabled ECG Monitor: Using the Cypress PSoC 6 MCU" [5], "Designing MCU-Based Lab Instruments: Using the Cypress PSoC 5LP MCU" [6], and "A Lock-In Amplifier: Based on a PSoC5LP SoC" [7]. Before getting into the details of this project, I'd like to quickly cover the unique features of the PSoC 5LP device.

PSOC 5LP SYSTEM ON A CHIP

Currently, there are MCUs available which contain high-quality mixed-signal peripherals. For example, Analog Devices has introduced arm-based MCUs with high resolution analog front ends and DACs, specifically targeted at chemical measurements.

I believe that the Cypress (now Infineon) PSoC 5LP MCU family is a viable choice for projects that require high-resolution analog functions along with digital functions. It contains high-resolution ADCs as well as DACs, op amps, comparators, and so forth. Other conventional MCUs contain a fixed number of peripheral functions and a multiplexer or cross-point switch to connect them to the external GPIO pins. The PSoC 5LP MCU also contains some pre-defined mixed-signal functions but also features a block of generic analog functions as well as generic digital functions. These can be configured by the user to perform many different tasks. The PSoC 5LP also has a comprehensive set of internal multiplexers to interconnect these blocks—to achieve a specific functionality, as well as bringing them out to external IO pins. [8]

Cypress' earlier PSoC 3/4 MCU families had many generic analog and digital blocks which could be combined to produce many different analog/digital functions. The tradeoff was that high-resolution/quality mixed-signal functions weren't available. In the PSoC 5LP MCU family, the focus is more on including a few high-quality analog/mixed-signal functions with a more direct signal path to the external MCU pins. The generic analog and digital function blocks still exist, but they are no longer needed to synthesize the higher-quality mixed signal blocks.

Before getting into the details of the PSoC 5LP MCU, I'd like to mention a few things that drew me to the Infineon PSoC family in general. Virtually all Arm 32-bit MCUs come in high-density packages, which means that working with the chips themselves is difficult unless you can do surface-mount technology reflow soldering on a custom-designed PCB. The PSoC 5LP device is available on a small development module which brings out all the device's pins to two 26-pin headers (with 0.1" spacing). Virtually all the I/O pins are uncommitted—Infineon has not included any other peripheral ICs on the board, which would consume I/O pins. The CY8CKIT-059 development board costs only $20 and comes with a snap-off programmer/debugger [8]. **Figure 2** shows the board with its programmer still attached.

The different PSoC devices are developed with the free PSoC Creator development IDE (a Windows-only application). I credit that as the reason why it loads and runs so quickly—it's not a Java application like other multi-platform development IDEs. I find it to be very user-friendly. I liken this IDE to a walled garden. Once you launch this application, you have everything you need to do development, at your fingertips, without leaving the application:

- A full C Compiler with context-sensitive syntax hints, like the Intellisense feature in Visual Studio
- Essentially, a graphical schematic capture program, where you drop the various analog/digital/system function blocks onto a work screen and interconnect them
- Right clicking on a function block allows you to enter a full configuration menu for the block or opens the datasheet for that block (in PDF format).
- A Pin page where you can examine and/or change the functions assigned to each device pin
- A clock page where you can graphically configure the many clocks present on the PSoC 5LP and which will generate the necessary start-up code.
- Flashing the device, as well as a full debugging capability, is built into the PSoC Creator app. This works seamlessly because all the PSoC development boards come with a CMSIS-DAP programmer built in (which can be snapped off in the case of the CY8CKIT-059 dev. board).

Somewhere between the introduction of the PSoC 5LP and the latest PSoC 6 MCUs, Infineon decided to introduce a radically different development toolchain called the Modus Toolbox. The philosophy behind Modus Toolbox is to simplify the C programming as much as possible by including various configuration "wizards" and "middleware" packages to minimize the complexity of programming the device.

While I have downloaded and played with the Modus Toolbox, I still prefer to use the PSoC Creator version 4.4 (probably the last Creator version that Infineon will introduce, given their promoting the newer Modus Toolbox).

Note that unlike most Arm-based 32-bit MCUs, the PSoC 5LP can run with a power supply of up to 5V. When powered that way, it has 5V logic levels. The CY8CKIT-059 prototyping board has no on-board regulator and therefore runs the PSoC 5LP at 5V when it gets its power from the attached programmer board (via USB). You can remove some jumpers and power it with 3.3V if that's what your other peripheral chips require. For this project, the 5V supply matched the needs of most of the other peripheral devices used.

THE 10.000MHZ TIME BASE

What I needed was a 10.000MHz VCXO module. I chose the Connor Winfield VKB61B2-010.0M module. It comes in a 6-pin package (7.5mm x 5.0mm), which I can easily handle. [9] It's meant to run on 3.3V, so I could not power it directly from the PSoC 5LP's power bus. However, the Air530 GPS module runs on 5V but contains an

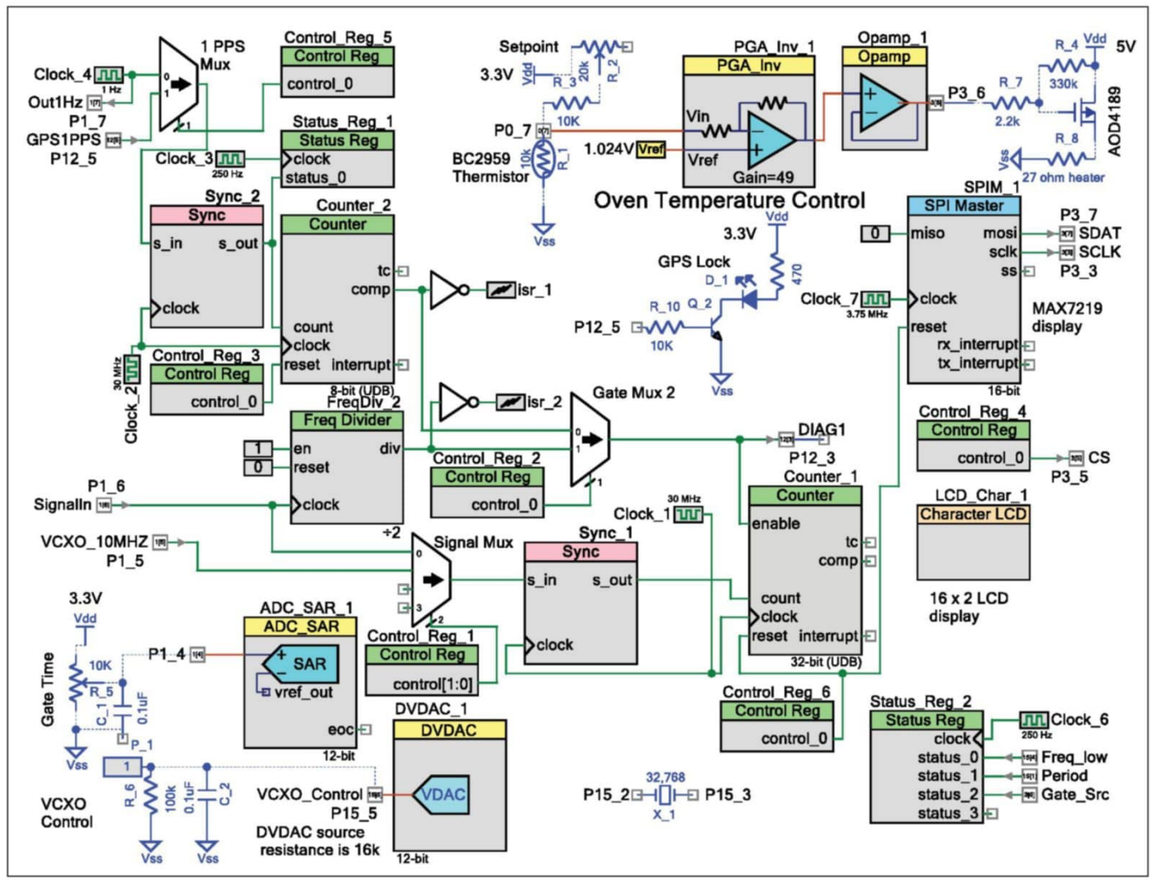


FIGURE 3
This is the "Top Design" page for this project, in the PSoC Creator IDE. All of the internal analog and digital blocks used in the project are shown here. External components are drawn in blue.

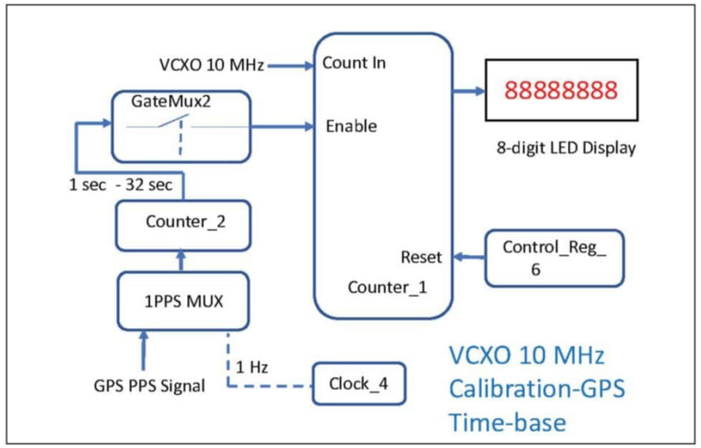


FIGURE 4
This is the block diagram of the circuit when in the VCXO calibration mode. The VXCO is calibrated using the 1PPS signal from the GPS module.

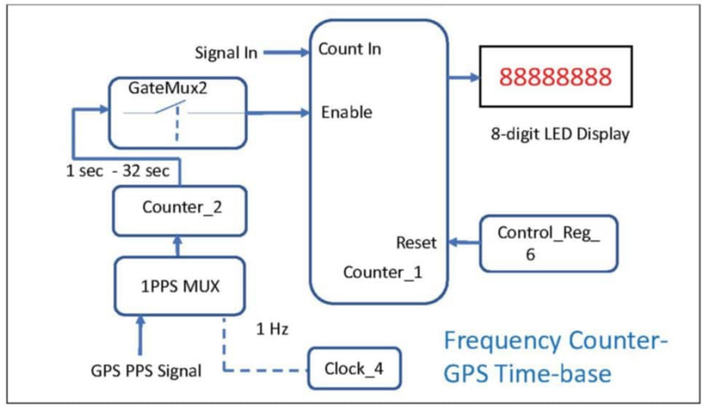


FIGURE 5
This is the block diagram of the circuit when in the frequency counter mode and using the GPS 1PPS gating signal.

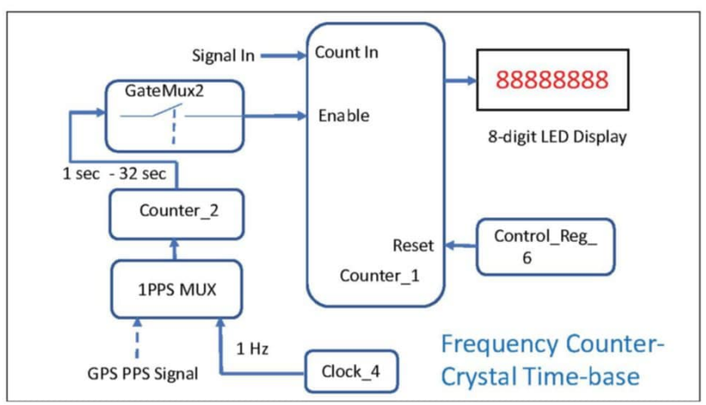


FIGURE 6
This shows the frequency counter mode, as in Figure 5, but using a 1 Hz clock signal provided by the PSoC5LP and a 32,768 Hz watch crystal (when no GPS signal is available).

internal 3.3V regulator and 3.3V is available on one of its pins. The VCXO's signal output is at LVCMOS levels (i.e., 3.3V). The PSoC 5LP, running at 5V, accepts this lower-level signal and clocks the internal circuitry properly with the 3.3V VCXO signal.

The frequency tolerance of the VKB61B2-010.0M is ±50ppm and it can be pulled ±100ppm over its control voltage range of 0–3.3V. The VKB61B2-010.0M's pull frequency is basically a linear function of its control voltage, but its monotonicity is only ±10%. I use a successive-approximation routine to trim the VCXO's output frequency to exactly 10.00000MHz by varying the voltage output of the PSoC 5LP's 12-bit DVDAC block, so the monotonicity isn't a concern.

The DVDAC's full-scale output voltage is 5V and I needed a 100kΩ shunt resistor, in combination with both the DVDAC's 16kΩ source resistance and the VKB61B2-010.0M's 100kΩ input resistance, to drop it down to the 3.3V range.

THE FREQUENCY/PERIOD COUNTER CIRCUITRY

Apart from the GPS module, the VCXO module, and some user interface components, almost all the necessary circuitry is contained within the PSoC 5LP. **Figure 3** is a PSoC 5LP schematic, as found in the "Top Design" page of the Creator 4.4 application. It contains most of the project's circuitry. Note that the PSoC Creator application allows you to draw in some external circuitry not contained in the PSoC 5LP itself. Such external circuitry is colored blue in this figure.

- Calibration of the VCXO using the 1PPS signal from the Air530 GPS module
- Measuring input signal frequency directly
- Measuring signal period
- Measuring low frequencies by counting the 10MHZ VCXO Time Base while using the input signal as the gate (like the bullet point above)

I'll explain each function using block diagrams which show only the function blocks necessary for that function.

Figure 4 shows the block diagram of the 10.000MHz VCXO Time Base Calibration function. The VCXO's 10MHz signal is fed into a 32-bit PSoC 5LP Counter_1. The counter's Enable line is fed from the GPS 1PPS signal, after being divided down by Counter_2. This gates Counter_1 for a precise interval in the 1-32 second range. The actual gate time starts off at 1 second at the beginning of the calibration routine and gets successively longer (for better resolution) as the successive approximation routine proceeds.

After the count has been accumulated over the gate interval, it is compared to 10,000,000Hz (factoring in the gate time). At the start of this routine, the PSoC 5LP's DVDAC_1 is set to mid-scale (2048) which places the control voltage of the VKB61B2-010.0M VCXO at midrange. Depending upon whether the VCXO's frequency is high or low, the DVDAC_1 is adjusted up or down and another measurement is taken. As mentioned above, the gate time is increased as the routine proceeds to increase the resolution of the frequency measurement.

This cycle is repeated until the VCXO's frequency is within 1Hz of 10,000,000Hz. I don't store the resulting VCXO's control voltage (in terms of its DVDAC value) in non-volatile memory, as it will vary somewhat over time, and the calibration should be performed at each power-up. If you are wondering what a Dithered Voltage Digital to Analog Converter (DVDAC) is, in the PSoC 5LP it is an 8-bit Voltage DAC which uses Dithering to achieve 12-bit resolution. The Dithering technique works well for a DAC application which only needs to produce slowly changing outputs.

Figure 5 shows the block diagram of the frequency counter function. It's like the last figure, except that the frequency to be measured is now what is counted by Counter_1. Counter_1 is still gated by Counter_2, which is fed by the precise 1PPS GPS signal. The user can vary the gate time in the 1 to 32 second range, with the longer gate times increasing the frequency resolution. The frequency is displayed on an 8-digit, 7-segment LED readout.

Figure 6 shows the block diagram of the frequency counter function, but in this case, the gating is provided by the 1Hz Clock4 signal. This is generated by the PSoC 5LP using its low-frequency oscillator circuit in conjunction with an external 32,768Hz watch crystal.

The accuracy of the frequency counter, in this configuration, is less than the GPS-derived gate configuration mentioned earlier and is meant to be used only if the GPS 1PPS signal is not available.

Figure 7 shows the block diagram of the period counter function. In this case the input signal is divided by 2 (to eliminate duty cycle from the measurement) and is used to enable Counter_1.

Counter_1 counts the 10,000,000Hz signal from the VCXO (which has been calibrated by the GPS 1PPS signal during the calibration routine). The period is derived by taking Counter_1's value and dividing that by 20,000,000 (to account for the division by 2).

Low-frequency signals are also measured using the same configuration but then taking the reciprocal of the period reading before displaying it. Because Counter_1 has a 32-bit depth, the maximum period measurable is (2^{32}) =4,294,967,296 divided by 20,000,000, or 214 seconds.

Please refer to Figure 3, the PSoC 5LP's internal function block diagram, containing some external circuitry in blue. Most of the functionality has been covered in the previous four block diagrams. However, there are some things that were not included in those block diagrams. While the configuration of the various counters was shown, in Figure 3 you will notice two blocks labelled Sync_x.

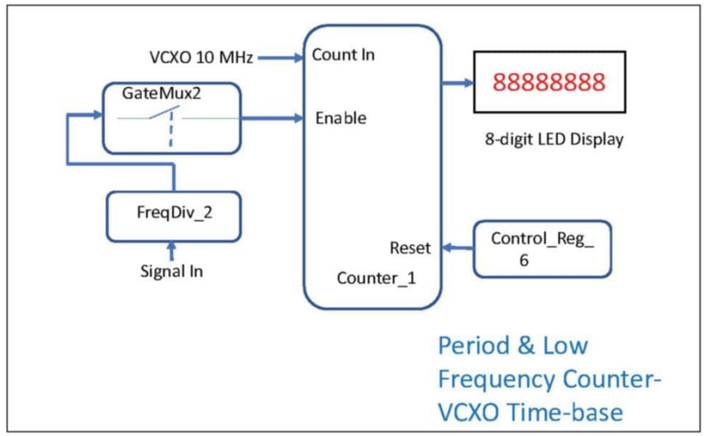


FIGURE 7
This shows the period/low-frequency measuring mode. In this case, the signal itself, divided by 2, acts as the gate. The 10MHz VCXO is counted by Counter_1, over the gate interval defined by the input signal.

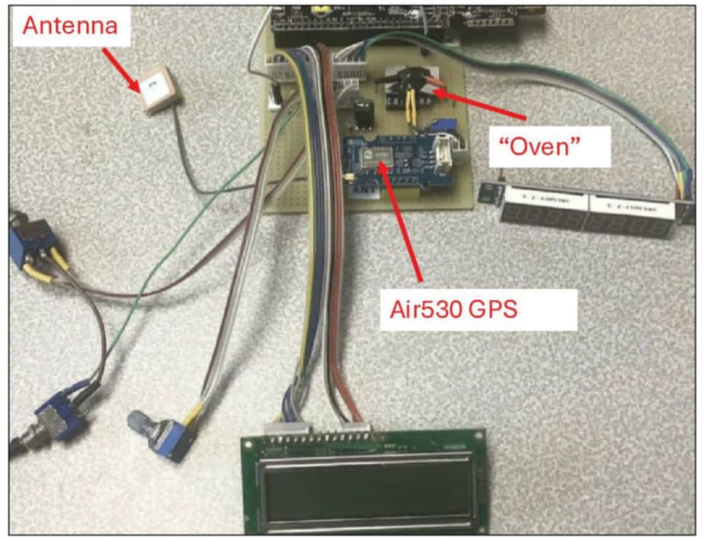


FIGURE 8
This is a photo of the complete circuit. You can see the crude "oven" for the VCXO, consisting of a heater/thermistor embedded in a blob of epoxy attached to a small aluminum plate. The VCXO is directly underneath this assembly.

COLUMNS

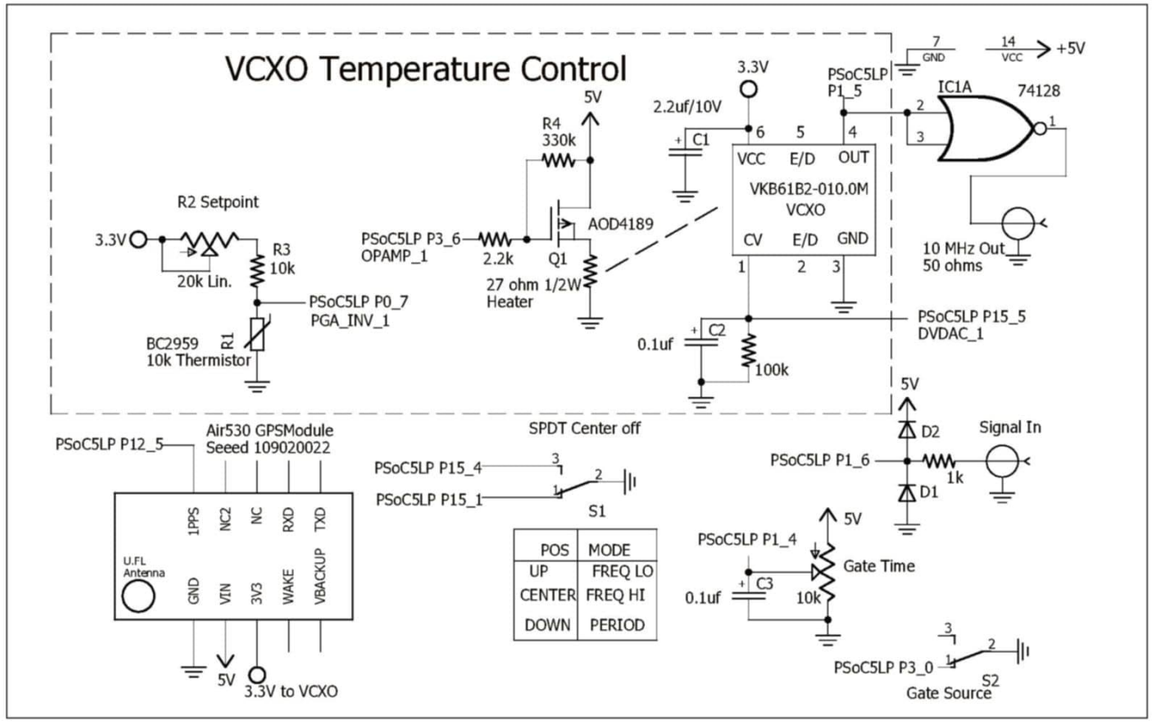


FIGURE 9
Although most of the active circuitry is shown on the PSoC5LP Top Design drawing (Figure 3), there are some additional circuits that are shown here.

These Sync blocks are needed to synchronize external, asynchronous signals with the internal clocks present in the PSoC 5LP. Each Sync block is fed with a 30MHz clock, which performs this synchronization.

The user-selectable Gate Time selection is handled by a 10kΩ pot, which is read by ADC_SAR_1 in the PSoC 5LP.

The PSoC 5LP's MCU runs from an internal RC clock, feeding a PLL. This is definitely not accurate enough to provide a 1Hz signal that might be needed when a GPS 1PPS signal is not available. An external 32,768Hz watch crystal is connected to PSoC 5LP pins P15_2 and P15_3, allowing the PSoC 5LP to generate a suitably accurate 1Hz time base for this purpose.

While I hesitate to call it an oven, I have added a heater and a closed loop control circuitry to keep the VKB61B2-010.0M VCXO at an adjustable elevated temperature, in the 30°C range. **Figure 8** shows all the circuitry and points out my rather crude "oven."

A 27Ω 1W resistor is used as the heater. This heater and a BC2959 thermistor are embedded in an epoxy blob fastened to a small aluminum plate. This plate is fitted on top of the VKB61B2-010.0M VCXO and held in place by the resistor leads themselves.

The BC2959 thermistor forms a part of a resistive divider from the 3.3V power supply. The divider output is fed into an inverting amplifier, PGA_Inv_1, which is an analog block contained in the PSoC 5LP.

The PGA's non-inverting input is fed by the PSoC 5LP's 1.024V reference source, so the PGA block is acting as a comparator with a gain of 49. The PGA output is fed to one of the four dedicated op amps contained in the PSoC 5LP.

I have configured it as a voltage follower, but it could have been easily re-configured as an op amp with gain, if necessary. The 27Ω resistor could draw almost 200mA at maximum power, given its 5V power source. While the PSoC 5LP's op amps can provide up to 25mA, an AOD4189 P-channel MOSFET was used to control the heater power.

The AOD 4189 is overkill here but was on hand and MOSFETS with a more suitable current rating are so small that I struggle to mount them.

The Setpoint pot, R_2, is adjusted, by trial and error, until the thermal block is warm to the touch, which would put it in the low 30°C range. All that I am trying to achieve here is to maintain the VCXO at a steady temperature somewhat above ambient—in my environment 30°C is sufficient.

To give an external indication that the GPS 1PPS signal is present, I power an LED via Q_2. This will blink once per second when a GPS signal has been captured.

The 8-digit LED display that I used is very common—found on Amazon and elsewhere—which uses a Maxim MAX7219 Display Controller. This controller uses a serial protocol similar to SPI.

The PSoC 5LP contains an SPI_Master function block. Generally, this SPI_Master block controls the SPI -CS signal in hardware, via its ss pin. However, this method doesn't work with the MAX7219, as it expects the -CS line to remain low for a complete command/data transfer. Therefore, I control the -CS line separately (port P3_5), under program control, in my MAX7219 driver routines.

While I wanted a reasonably large LED readout for the actual numeric frequency/period values, in operation there are numerous times when a descriptive text message is useful. Therefore, I added a small 16-char x 2-line alphanumeric LCD display. Although I generally use I²C versions of these LCD displays, the PSoC Creator IDE contains a library that interfaces with these displays using the older 4-bit parallel protocol, so that is what I used.

To a limited extent, the Creator 4.4 application allows the user to add some external components to its Top Design page (such as in Figure 3), which I have done. However, there is some additional circuitry which wouldn't fit, which can be found in **Figure 9**. The VCXO circuit is shown here, as well as the connections to the Air530 GPS module. The temperature controller circuitry is also shown here, although it is also contained in Figure 3 as well.

The user interface is shown in this figure. The Gate Time pot was described earlier and is also shown in Figure 3. The three measurement modes are handled by a SPST switch with a center-off position. S2 is the Gate Source selector. In the absence of a GPS 1PPS signal, an internal, crystal-controlled 1Hz time base can be selected. In this instance, the accuracy will not be nearly as good as that provided when a GPS signal is available.

I didn't allow for a separate user switch to allow the VCXO to be recalibrated after the unit had been running for an extended period. When the unit is powered up, the Gate Source selector switch will be polled.

If it is in the GPS position, a calibration of the VKB61B2-010.0M VCXO will be performed. Otherwise, this calibration routine will be skipped, since the user has presumably determined that there is no GPS signal available, which would cause the VCXO calibration routine to just hang.

A VCXO recalibration can be performed at any time by merely cycling the power to the unit. While not visible in the photo (Figure 8), the Air530 GPS module contains a 3V coin cell for backup power (mounted on the bottom of the module). This allows for a short "warm start" of the GPS module, after which the VCXO calibration routine takes less than 1 minute to perform.

I added a 74128 NOR Gate to buffer the output of the VCXO before it is fed to a BNC connector to provide an accurate 10.0000MHz clock to external devices. This particular TTL chip is designed to drive 50Ω loads.

The signal input to the frequency counter is passed through a 1k resistor and on to diodes D1,D2 which prevent the input signal from damaging the PSoC 5LP's port P1_6.

ITERATION IMPROVEMENTS

Looking back at my first iteration of this project, it was satisfying to have been able to eliminate all the TTL glue chips that were present in the earlier design, replacing them all with function blocks in the PSoC 5LP. Using the Connor Winfield VCXO certainly allowed the Air530 GPS module to perform a lot better in an indoor setting, with no interference due to RF leakage from the Si5351 used in the first iteration.

It seems like Infineon has moved on from the PSoC 5LP system on a chip in favor of newer models like the PSoC6. However, in terms of the sheer variety of digital/analog functions that can be custom configured in the PSoC 5LP, the newer PSoC 6 family seems to be a step backward.

I find this to be a shame, particularly since I find designing with the PSoC 5LP, using the PSoC Creator 4.4 IDE, such a pleasure.

Additional materials from the author are available at:
www.circuitcellar.com/article-materials
References [1] to [5] as marked in the article can be found there.

RESOURCES

Infineon Technologies | www.infineon.com

Seeed Technology | www.seeedstudio.com

Analog Devices | www.analog.com

ABOUT THE AUTHOR

Brian Millier runs Computer Interface Consultants. He was an instrumentation engineer in the Department of Chemistry at Dalhousie University (Halifax, NS, Canada) for 30 years.

Embedded System Essentials

Using a Desktop Laser Engraver to Probe into Chip Silicon

By
Colin O'Flynn

Laser fault injection is a common method of attacking devices, but sometimes we want to first understand what a device looks like or what certain blocks on the surface of the device die are. This article shows how you can use a low-cost laser engraver to map out these blocks, even without more advanced fault injection equipment.

This article will look at low-cost ways you can see into a chip and explore its functionality. Normally when we talk about topics such as imaging the surface of a chip, we probably imagine more complex processes such as dissolving the silicon encapsulation in acid and using expensive microscopes. Luckily, however, there is a quick hack: using a common type of package called Wafer Level Chip Scale Package (WLCSP).

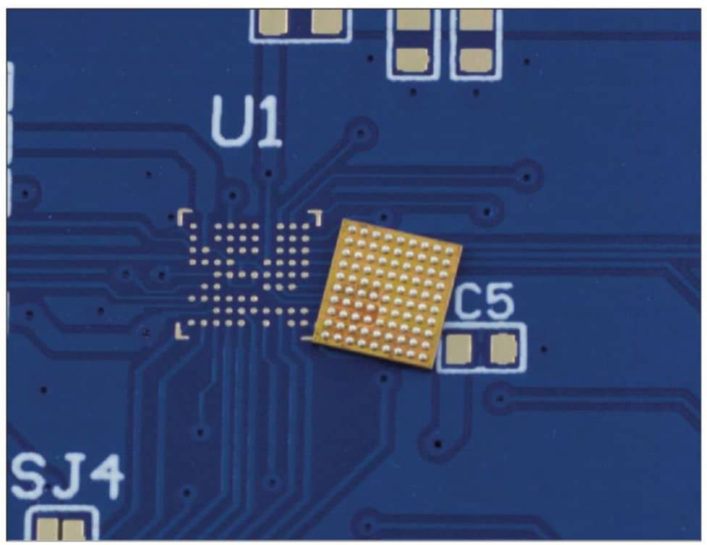


FIGURE 1
WLCSP device on a PCB

WAFER-LEVEL PACKAGING

WLCSP are a type of package which means there is very little "extra" silicon besides the IC die itself. A photo of the underside of such an IC (a STM32 chip) is shown in **Figure 1**. It's important to remember that many ICs come in various packages, and typically the actual die is the same. We still find information about the WLCSP useful for other packages, as it can help us if we are trying attacks that require understanding the position of certain parts of the chip.

I first discussed this when performing security analysis in my article "IC Packaging and Security: Do They Affect One Another?" (*Circuit Cellar* 362, September, 2020) [1]. In that article, I mentioned we could use that to probe the backside of the ICs dies. This time, I'm going to do everything without touching the backside. And in fact, we're not even going to be attempting to perform an attack, we're just going to be looking at how we can measure important information such as understanding the layout of a target IC.

One thing I'm going to show you is that it can help you to understand sensitive areas of the chip you want to *avoid* attacking. If using something that can be positioned, such as electromagnetic fault injection (EMFI) or laser fault injection (LFI), you have control over what area of the chip you target. A common problem when using these techniques is the possibility of damaging the device, so understanding the layout is important to know which areas you may wish to avoid targeting.

LOOKING THROUGH THE BACKSIDE

Back in 2023, Andrew "bunnie" Huang published a blog post titled "Infra-Red, *In Situ* (IRIS) Inspection of Silicon," which pointed out that most low-cost camera-based microscopes (the type used for looking at PCBs for soldering inspection) could actually see through the silicon backside of WLCSP devices [2]. This is because silicon is relatively transparent in the 1000-1100nm wavelength; this is, in fact, normally exploited with LFI, which use 1064nm wavelength lasers.

But most CMOS image sensors are just sensitive enough to that wavelength so you can illuminate the backside of an IC with 1064nm LED (easily available from your favorite parts supplier). Under normal conditions, we wouldn't want the sensor to see the IR light, so most will have an IR filter in the optical path. You will need to remove the IR filter to take photos, as the IR filter will block this wavelength. Save the IR filter as we'll reuse it momentarily!

A simple photo taken from the microscope is shown in **Figure 2**. Different angles will illuminate different areas of the chip, and by compositing a few photos you can get complete images. The Figure 2 example wasn't at maximum zoom level; I've purposely included parts of the PCB as I want to use that to understand where pin 1 is located relative to the PCB.

If you do try this, remember that many WLCSP have a covering on the backside to prevent light from entering (for example, a strong camera flash could cause the device to reset). More information on this can be found in my previously mentioned *Circuit Cellar* article.

Using this technique, I've shown you how you can optically see what is inside your device. Besides some obvious blocks like big arrays that must be memory, what else can we learn? With a low-cost laser engraver, we can see which areas of the chip might be more sensitive.

BURNING YOUR WAFER

The second part of my imaging will use an off-the-shelf laser engraver, specifically the XTools F1. You might expect I'd be using this for LFI, but this particular unit has limited use for that, though you can still conduct some attacks. A few other people have shown how to interface with it; I was first put onto the idea after seeing it in Fraktal's Laser Fault Injection article [3].

The reason that it isn't very useful for LFI is that while it has a 1064nm laser that will pass through the die, the laser cannot easily be triggered at a specific point in time. The laser diode itself automatically creates the

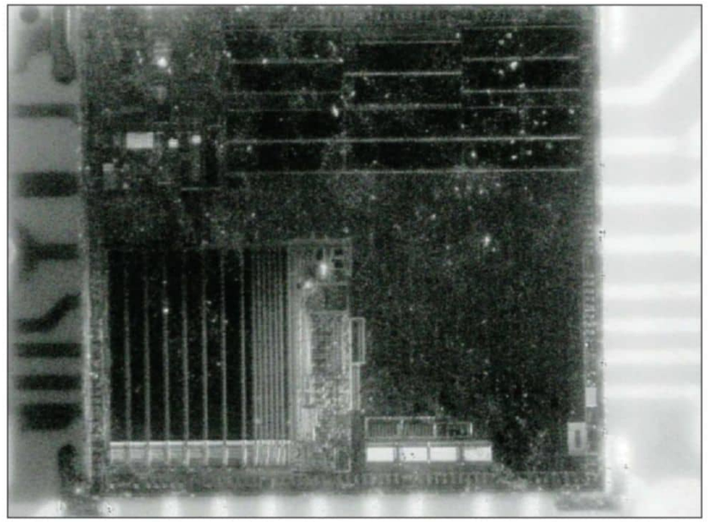


FIGURE 2
The STM32F405OG device on the PCB; combing multiple images will improve the overall contrast.

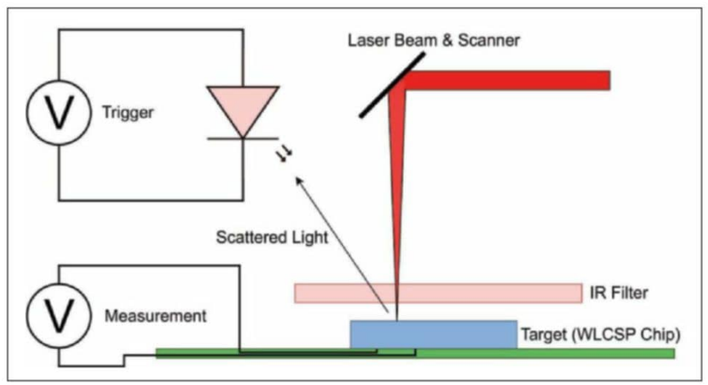


FIGURE 3
As laser pulses are inserted into the device, a scope probe (shown as the "V" voltmeter symbol) measures the magnitude of the resulting VCC spike.

COLUMNS

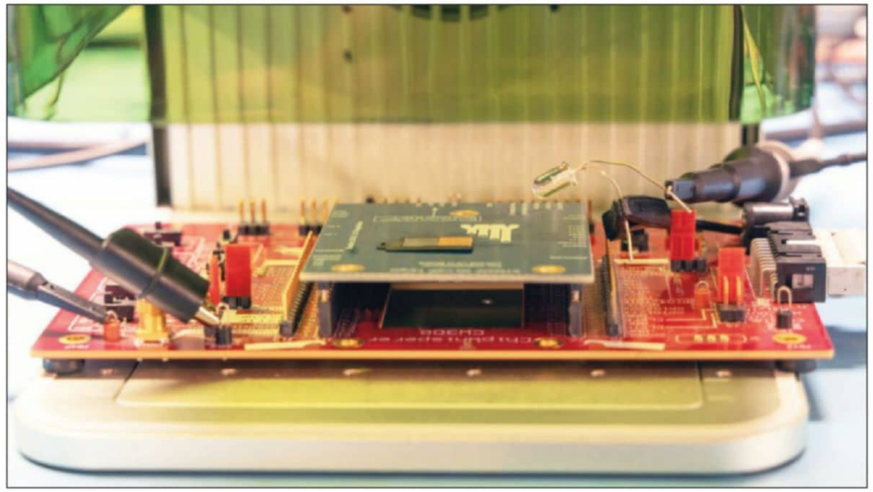


FIGURE 4
The physical setup shows the laser shield (translucent green shield at top) raised for the photo. Oscilloscope probes measure the VCC spike resulting from a laser pulse inserted at specific points.

pulses of a (roughly) consistent pulse power. In order to scale the power output for laser engraving, the period between these pulses is adjusted, but each pulse is the same power.

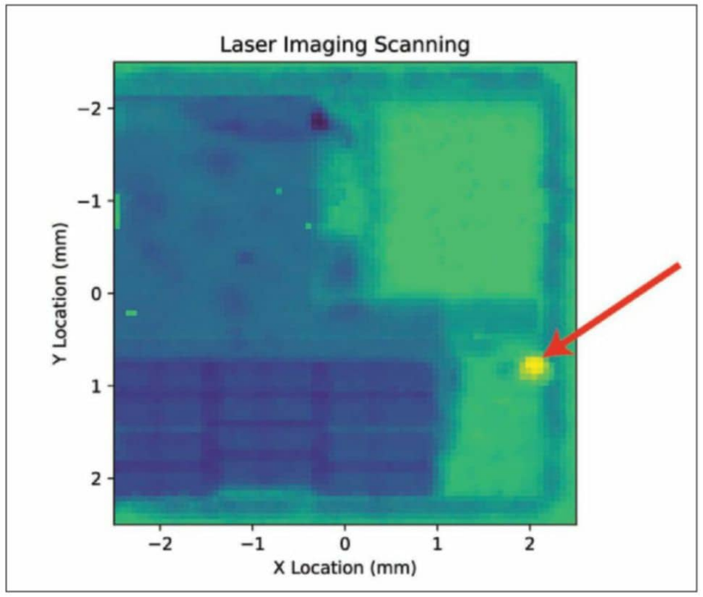


FIGURE 5
The magnitude of a VCC power spike at each laser spot injection location is turned into a color; bright yellow is the highest.

Additional materials from the author are available at:
www.circuitcellar.com/article-materials
References [1] to [4] as marked in the article can be found there.

Resources

STMIcroelectronics | www.st.com

Typically, the 1064nm laser diodes used to create pulses of light are a "Q-switched" architecture, where laser power builds up in a lasing cavity until a pulse of laser power is emitted by switching the Q-switch medium. This can be an active (controlled or triggered) switch or a passive switch that is based on the pulse power. Our low-cost engraver uses a passive switch, which means we cannot control either the exact timing or the power of the pulse; the laser pulse output is set by the physical design of the laser.

The laser engraver adjusts the period between the pulses by controlling power to the laser diode. This simply delays the laser turning on to full power but does not give us very accurate timing.

However, the laser engraver does have a nice set of mirrors (a galvanometer scanner), which can position the laser onto different locations on the chip very quickly. This means we can position the laser pulses at various points of the IC die, which is again transparent to the 1064nm of the engraving laser. So, even though we can't set accurate timing information, we can set accurate position information.

Next, we'll attach an oscilloscope to the VCC line of the IC while I scan across the surface of the chip. As I insert laser pulses into the chip surface, we'll find they couple into different levels of power in the VCC pin. **Figure 3** shows a diagram of this setup. You might notice I kept the IR cut filter from our microscope, as it reduces the power of the laser. The default power is too high and would often damage the chip, so using the IR cut filter keeps a more usable level of laser power reaching the surface of the chip.

To trigger on the pulse being inserted, I'll use the same 1064nm LED from earlier as a photodiode. LEDs can be used as photodiodes; they aren't as sensitive as one designed for it, but with the strong laser pulse it's sufficient to use as-is. This photodiode will sense the reflected pulse from the surface of the die, and I can use that to trigger my oscilloscope.

Figure 4 shows a photo of my setup, with the laser cut filter on top of the WLCSP to avoid overloading the die with too powerful of a pulse. Other materials work, but if you already have an IR filter, why not use that?

LASER SCAN RESULTS

The results of one example scan are given in **Figure 5**. Comparing Figure 2 and Figure 5 show clear common patterns—especially the blocks of what one assumes is memory in the lower left corner. You can see right away how this provides a visual method of imaging the device, but so far I'm not learning anything more than I did with my cheap microscope. Let's dive into what we can do to understand the actual functionality of various blocks.

With the oscilloscope connected to the VCC line, one interesting area is the bright yellow dot highlighted with a red arrow in Figure 5. This is almost certainly the internal voltage regulator of the chip. This makes sense, as you can imagine that inserting laser pulses into this area will more readily couple onto the VCC rail that I'm measuring.

I found that this area is also very sensitive to damage if attempting LFI. My earlier warning that I could easily kill the chip without the IR cut filter in place was primarily due to hitting this voltage regulator with the laser. Most likely, I was actually causing the voltage to go so far out of spec that it could damage the device, or causing other damage resulting in too high of a current draw and preventing the device from powering up normally. Using this low-power scan first, I can decide on areas I should avoid inserting faults into.

By comparison, if I scan the SRAM cells with the laser, they are much less sensitive to permanent damage. While I originally said I wasn't going to use this low-cost engraver for fault injection, it can be used for attacks where timing isn't critical.

Pointing the laser at the SRAM cells causes corruption in the cells, and I can remove my IR cut filter during these tests. I limit the laser probing to what I suspect is the SRAM cells (avoiding the voltage regulator). I then download a pattern using a JTAG programmer to the SRAM cells and read out the pattern after each fault injection attempt.

In **Figure 6** the red crosses indicate the location of corruption that I observe—note this overlays with the blocks that I suspected were indeed SRAM cells. By performing this test with other areas of the chip, I can help map out exactly where items such as specific hardware blocks are located.

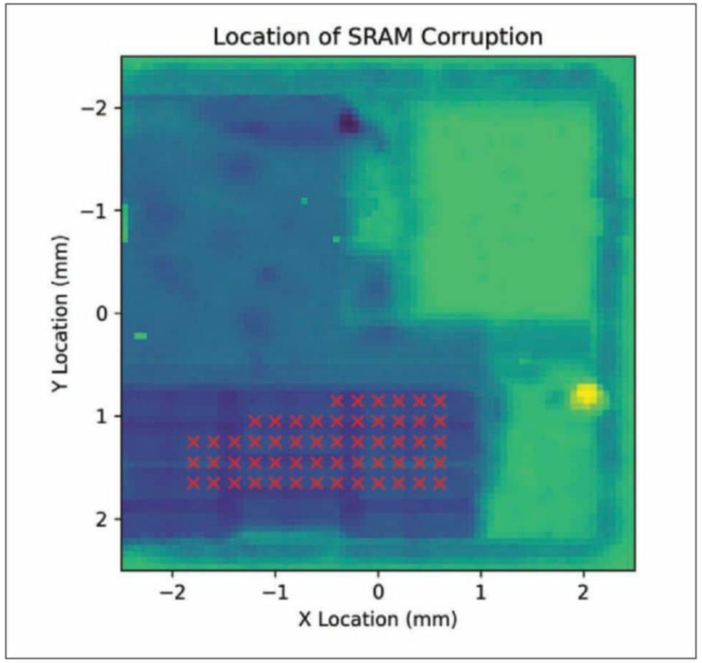


FIGURE 6
Overlaying the location of SRAM corruption (as measured by checking SRAM contents) is marked with an X, allowing discovery of specific SRAM address locations on the die.

A laser with more controllable parameters would let me perform more advanced reverse engineering, but you can see even a low-cost laser allows me to get useful information out of it. And if you already have a desktop laser engraver, this might be something you can experiment with for free.

Hopefully this article presents you with some ideas on how you can use low-cost tools. More information, including some of my scripts and additional examples, can be found on the companion GITHub webpage, which is located on the *Circuit Cellar* Article Materials and Resources webpage [4]. If you have any success let me know what you managed to image!

ABOUT THE AUTHOR

Colin O'Flynn (colin@oflynn.com) has been building and breaking electronic devices for many years. He is CTO of NewAE Technology based in Halifax, NS, Canada and is currently an assistant professor at Dalhousie University. Some of his work is posted on his website at www.colinoflynn.com.

Vishay Intertechnology Power Metal Strip Resistor Delivers Power to 5 W in Compact 1206 Case Size

Vishay Intertechnology, Inc. introduced a new surface-mount Power Metal Strip current sense resistor that combines a high power rating up to 5W, TCR down to ± 75ppm/°C, and extremely low resistance values down to 0.3mΩ in the compact 1206 case size.

With power ratings 20 times higher than standard resistors in the 1206 footprint, the Vishay Dale WSLF1206 provides a power density of >650W/in², allowing designers to save board space in high power circuits by utilizing the smallest resistor possible without the need to parallel resistance elements. The device's low TCR provides stable and accurate current measurement across a wide temperature range from -65°C to +170°C, while its low resistance values minimize power losses and improve end product efficiency.

The WSLF1206's advanced construction incorporates a solid metal manganese-copper, nickel-chromium aluminum alloy resistive element with low TCR (<20ppm/°C). This results in a higher power resistor in a smaller size that maintains the superior electrical characteristics of the Power Metal Strip construction. A proprietary processing technique produces resistance values ranging from 0.3mΩ to 3mΩ, with tolerances of ±1.0% and ±5.0%. The device provides low thermal EMF (<3µV/°C) and very low inductance (<5nH) to minimize signal distortion.

The resistor released today is ideal for all types of current sensing and pulse applications, including switching and linear power supplies, power amplifiers, shunts, power inverters, and battery management. Typical applications will include industrial motor drives, robotics, power tools, HVAC systems, and UPS; graphics, data storage, and VRMs for supercomputers; and consumer appliances. The device is sulfur-resistant, RoHS-compliant, and halogen-free.

Vishay Intertechnology | vishay.com

Bourns Expands Semi-Shielded Power Inductor Portfolio with New Series Featuring Higher Maximum Inductance Values

Bourns, Inc., a leading manufacturer and supplier of electronic components for power, protection, and sensing solutions, announced the expansion of its semi-shielded power inductors. Bourns has added five new product series to the existing Bourns SRN-BTA family. These new products offer higher maximum inductor values—with some models doubling previous limits—while delivering lower magnetic field radiation. This provides a cost-effective alternative to fully shielded ferrite-based inductors. The semi-shielded construction combines the advantages of both non-shielded and shielded inductor designs, offering a robust and reliable solution for demanding power applications.

The new additions to the SRN-BTA family enhance product reliability with bottom-soldered lead-wires, providing increased mechanical strength and stability. These models are well-suited for applications in automotive systems, DC-DC converters, and power supplies across consumer, industrial and telecom electronics—particularly in applications requiring higher power inductors. These advancements deliver multiple benefits, including improved magnetic performance and greater efficiency.

The Bourns SRN2010/2510/3010/3015/4018BTA Series Power Inductors are available now through Bourns' authorized distribution partners. These models are AEC-Q200 compliant and automotive grade, meeting the rigorous standards required for automotive applications. These products are manufactured within a Bourns' IATF 16949 certified facility.

Bourns, Inc | bourns.com

STATEMENT REQUIRED BY THE ACT OF AUGUST 12, 1970, TITLE 39, UNITED STATES CODE SHOWING THE OWNERSHIP, MANAGEMENT AND COPY CIRCULATION OF CIRCUIT CELLAR. Published monthly at 650 Dairy Farm Rd, Red Oak, VA 23964. Annual subscription price is $50.00. Publisher: KC Prescott. The owner is KCK Media Corp., 650 Dairy Farm Rd, Red Oak, VA 23964. The names and addresses of stockholders holding one percent or more of the total amount of stock are: KC Prescott, 650 Dairy Farm Rd., Red Oak VA 23964. EXTENT AND NATURE OF CIRCULATION: Average number of printed copies of each issue published during the preceding twelve months; (A) total number of copies printed, 2285; (B.2) paid/requested mail print subscriptions, 898; (B.3) sales through dealers and carriers, street vendors and counter sales, 892; (B.4) paid/requested copies distributed by other mail classes, 45; (C) total paid/requested print circulation, 2063; (D.4) Nonrequested copies distributed outside the mail, 208; (E) total nonrequested distribution (sum of D.1 & D.4), 208; (F) total distribution (sum of C & E), 2271 (G) copies not distributed (office use, leftover, unaccounted, spoiled after printing, returns from news agents), 14 (H) total (sum of F & G), 2285. Percent Paid Requested: 91% (A) total number of paid Electronic Copies, 5627; (B) Total Paid Print & Paid Electronic Copies 7690 (C) Total Print Distributions and Paid Electronic Copies 7898 (D) Percent Paid: 97% Actual number of copies of a single issue published nearest to filing date: (A) total number of copies printed, 2357; (B.2) paid/requested mail print subscriptions, 874; (B.3) sales through dealers and carriers, street vendors and counter sales, 930; (B.4) paid/requested copies distributed by other mail classes, 43; (C) total paid/requested circulation, 2051 (D.4) Nonrequested copies distributed outside the mail, 300; (E) total nonrequested distribution (sum of D.1 & D.4), 300; (F) total distribution (sum of C & E), 2351 (G) copies not distributed (office use, leftover, unaccounted, spoiled after printing, returns from news agents), 6; (H) total (sum of F & G), 2357. Percent Paid Requested 87%. (A) total number of paid Electronic Copies, 5648; (B) Total Paid Print & Paid Electronic Copies 7699 (C) Total Print Distributions and Paid Electronic Copies 7999 Percent Paid: 96% I certify that the statements made by me above are correct and complete. KC Prescott Publisher.

STMicroelectronics' New Automotive Linear Voltage Regulator Preserves Battery Energy in Challenging Conditions

The TL99VR03 300mA low-dropout (LDO) regulator provides resilient and efficient power, with a wide input-voltage range and very low quiescent current consumption, only 3.5µA at no load. The IC has an enable pin for turning the regulator off, which reduces the idle current to 800nA, and integrates soft-start circuitry to limit current during power-up and fault recovery.

Qualified to AEC-Q100, the L99VR03 powers circuitry such as automotive microcontroller systems, logic ICs, and sensors in body-control modules, telematics controllers, lighting controllers, and head units. Additional applications include providing sustaining power for slow ramp-up systems. The device has a wide operating temperature range, from -40°C to 175°C, allowing deployment under the hood in combustion engine vehicles and in the power modules of EV drivetrains.

ST's L99VR03 LDOs are available with a fixed 3.3V or 5V output and operate with up to 40V at the input. The output is accurate to within ±2% in all operating conditions and the wide input-voltage range ensures the output remains stable throughout events such as cold cranking and load dump. Switching noise rejection is greater than 60dB at 1kHz and the regulator requires only 2.2µF output capacitance to ensure stability.

The L99VR03 is in production now in a choice of two thermally enhanced package options, including a 3mm x 2mm bottom-terminated VFDFN8L with wettable flanks that facilitate quality inspection. There is also a leaded PowerSSO12 package with a 4.9mm x 3.9mm body that contains a large heat spreader to maximize dissipation.

ST | st.com

Saelig Announces the Cambrionix ThunderSync5-C16 PD—the World's Fastest Enterprise USB Hub

Saelig Co., Inc. has announced the Cambrionix ThunderSync5-C16 PD, claimed to be the world's fastest, most reliable, and intelligent multi-port USB hub, aimed at addressing the frustrations of mobile device testing and refurbishment. It features a technology first—a built-in one-click DFU mode—as well as full remote management, that will greatly enhance high-volume mobile device processing and test workflows. The hub guarantees a minimum of 1.25Gbps and up to 5Gbps per port, maximizing performance for high-speed device processing and testing applications. Key performance advantages include:

- 7x faster data speeds than alternative USB hubs
- 8x faster charging compared to alternative USB hubs
- 25% faster data transfer than Cambrionix's ThunderSync3
- 96 devices processed at once through daisy-chaining

Currently software and device testing or refurbishing of hundreds of mobile devices can take hours of repetitive work, due to the need for manual set up and the impact of test failures triggered by power or USB endpoint limitation issues. The ThunderSync5-C16 PD eliminates device dropouts caused by USB endpoint limitations. The ThunderSync5-C16 PD greatly reduces test time by automating zero-touch test configurations and speeding data throughput by up to 7x, saving hours for IT teams, software testers, educators, and refurbishers.

The ThunderSync5-C16 PD hub incorporates intelligent functionality designed for enterprise-grade device management through the Cambrionix Connect Premium license and provides:

- One-click multi-device DFU mode for Apple silicon Macs and iOS devices with USB-C ports
- Flexible Power Delivery Management—Choose the maximum power output per port (60/ 45/30/27/15/4.5W)
- Schedule port and target charge mode—Turn ports on/off at specific times, set % charge levels on devices
- Remote hub management through Cambrionix Connect software
- Custom device processing workflows via programmable ports
- API and CLI integration through Cambrionix Toolkit
- Multi-mounting options including desktop, surface, rack, and DIN configurations

The hub also features advanced USB Power Delivery (PD) with a shared 285W power output, supporting flexible, configurable per-port power profiles ranging from 4.5W to 60W. This intelligent power management system adapts to individual device requirements, optimizing charging efficiency across diverse device portfolios to ensure test stability during heavy power demand and long-term viability of device batteries.

The ThunderSync5-C16 PD's performance over other hubs in real-world iOS device restoration scenarios will make it a valuable resource for app software QA laboratories requiring high-volume device testing, cloud device farms, professional test houses conducting comprehensive device validation, and OS and app releases/updates needing rigorous testing on real devices prior to deployment to speed delivery of bug-free new features. Millions of smart phones, tablets and laptops traded in or recycled every week need to be charged, erased, updated, configured and tested, and device processing speed is key to maximizing profitability. Educational institutions, healthcare organizations, and retail, transport, and government sectors will all benefit from employing the ThunderSync5-C16 PD Hub.

Saelig Company, Inc. | saelig.com

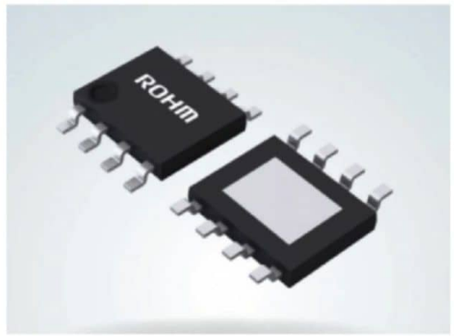
NEW PRODUCT SUBMISSIONS— E-mail: product-editor@circuitcellar.com

PRODUCT NEWS

ROHM Develops New Smart Switches Optimized for Zonal Controllers

ROHM Semiconductor announced the release of six new high-side smart switches (IPDs: Intelligent Power Devices) featuring highly accurate current sensing and low ON resistances ranging from 9mΩ to 180mΩ. These devices are designed to protect loads and subsystems from abnormalities such as overcurrent, overvoltage, and overtemperature, ensuring reliable operation and safeguarding sensitive components in automotive lighting and body control (i.e., door locks and power windows). Comprehensive diagnostic functions including open load and reverse battery detection further enhance system safety and reliability.

Modern zonal controllers face the complex task of managing numerous loads simultaneously, yet conventional smart switches often lack sufficient drive capability for high-capacitance applications. ROHM's new series addresses this limitation, delivering exceptional capacitive load driving performance along with low ON resistance and robust inductive energy clamp characteristics. By commercializing high-performance smart switches tailored to zonal controller requirements, ROHM is contributing to automotive electrification and the elimination of mechanical fuses.



The BV1HBxxxEFJ series (BV1HB008EFJ-C, BV1HB012EFJ-C, BV1HB020EFJ-C, BV1HB040EFJ-C, BV1HB090EFJ-C, BV1HB180EFJ-C) delivers exceptional high-capacitance load driving capability at the critical interface between zonal controllers and output loads (including ECUs). Proprietary cutting-edge process technology eliminates the traditional trade-off between low ON resistance and high inductive energy clamping, achieving an optimal balance across drive capability, ON resistance, and energy tolerance. This results in improved system safety, efficiency, and reliability.

At the same time, class-leading* current sensing precision (±5%) provides superior harness and fault protection, while the compact, high heat dissipation HTSOP-J8 package ensures excellent design versatility.

Going forward, ROHM remains committed to advancing safety, security, and energy efficiency in the automotive sector through the ongoing development of high reliability, high performance devices.

ROHM Semiconductor | rohm.com

*ROHM study on high-side Smart Switches—September 2025

Samtec Launches 800-Position AcceleRate HP High-Performance Array Connectors

Samtec, Inc., a global leader in high-performance interconnect solutions and a service leader in the industry, announces the expansion of its AcceleRate HP product line with the release of 800-position APM6 and APF6 high-performance array connectors available in a low-profile 5mm stack height.

Engineered for next-generation demands, the new AcceleRate HP connectors deliver exceptional signal integrity and density, supporting data rates up to 112Gbps PAM4 on a 0.635mm pitch. These connectors are optimized for high-throughput applications such as High-Performance Computing (HPC), Artificial Intelligence (AI), storage, and networking.

Key features include:

- Protocol compatibility with PCIe 6.0, CXL 3.2, and 100 GbE
- 800-position configuration (8 rows × 100 positions) in a dense footprint of 68.62mm × 18.20mm (2.701" × 0.717")
- 5mm mated stack height, with a 10mm version planned for Q2 2026
- 5Ω impedance
- 2A max current rating (one pin powered per row), and 150VAC (212VDC) max voltage rating
- Open pin field array design for maximum grounding and routing flexibility

The APM6 and APF6 connectors utilize Samtec's Solder Column Termination, enhancing structural integrity. This IPC Class 3 compliant method is ideal for dense, high-speed interconnects and offers superior solder joint reliability and excellent insertion loss and return loss performance.

"These new 800-position connectors represent a significant advancement in high-density, high-speed interconnect technology," said Eric Mings, Samtec's High-Speed Board-to-Board Product Manager. "They're designed to meet the rigorous demands of today's data-intensive systems while maintaining reliability."

Samtec, Inc | samtec.com

NEW PRODUCT SUBMISSIONS— E-mail: product-editor@circuitcellar.com

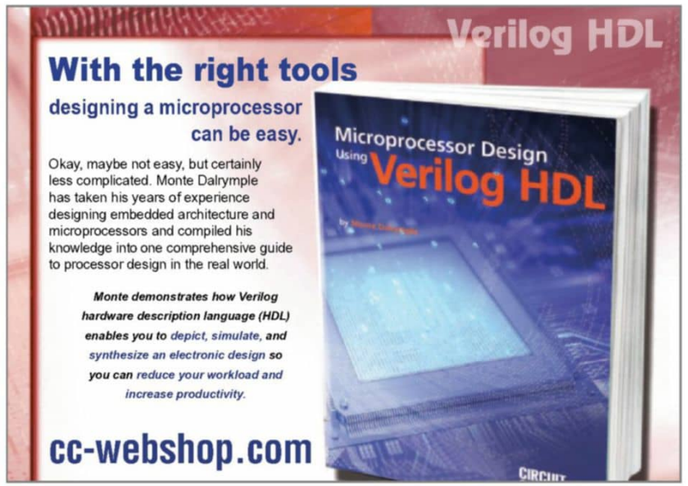
IDEA BOX

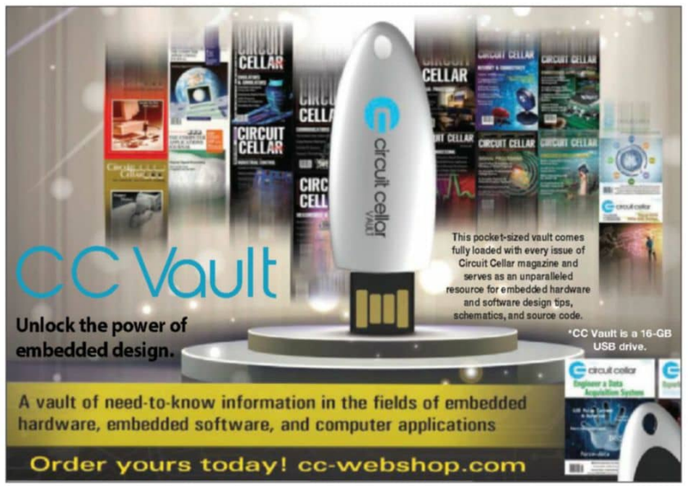
The Directory of PRODUCTS & SERVICES

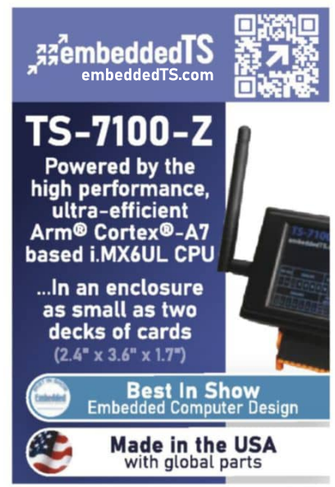
AD FORMAT:
Advertisers must furnish digital files that meet our specifications (www.circuitcellar.com/mediakit).

All text and other elements MUST fit within a 2" x 3" format.
E-mail adcopy@circuitcellar.com with your file.

For current rates, deadlines, and more information contact Hugh Heinsohn at 757-525-3677 or Hugh@circuitcellar.com


Verilog HDL
With the right tools
designing a microprocessor can be easy.
Okay, maybe not easy, but certainly less complicated. Monte Dalrymple has taken his years of experience designing embedded architecture and microprocessors and compiled his knowledge into one comprehensive guide to processor design in the real world.
Monte demonstrates how Verilog hardware description language (HDL) enables you to depict, simulate, and synthesize an electronic design so you can reduce your workload and increase productivity.
cc-webshop.com
Microprocessor Design Using Verilog HDL


CC Vault
Unlock the power of embedded design.
This pocket-sized vault comes fully loaded with every issue of Circuit Cellar magazine and serves as an unparalleled resource for embedded hardware and software design tips, schematics, and source code.
*CC Vault is a 16-GB USB drive.
A vault of need-to-know information in the fields of embedded hardware, embedded software, and computer applications
Order yours today! cc-webshop.com


embeddedTS
embeddedTS.com
TS-7100-Z
Powered by the high performance, ultra-efficient Arm® Cortex®-A7 based i.MX6UL CPU
...In an enclosure as small as two decks of cards
(2.4" x 3.6" x 1.7")
Best In Show
Embedded Computer Design
Made in the USA
with global parts

TEST YOUR EQ

Contributed by David Tweed

Question 1 —

Let's talk about noise! There are different types of noise that might be present in a system, and it's important to understand how to deal with them.

For example, analog sensors and other types of active devices will often have AWGN, or Additive White Gaussian Noise, at their outputs. Any sort of analog-to-digital converter will add quantization noise to the data. What is the key difference between these two types of noise?

Question 2 —

Signal-to-noise ratios are most usefully described as power ratios. How does one characterize the power levels for both AWGN and quantization noise?

Question 3 —

When you have multiple sources of noise in a system, how can you characterize their combined effect on the overall system performance?

Question 4 —

Broadband analog sensors and other active devices often specify their noise levels in units of "microvolts per root-Hertz" (μV/√Hz) or "nanoamps per root-Hertz" (nA/√Hz). Where does this strange unit come from, and how do you use it?

The Future of Embedded Systems and AI

AI in Embedded Systems

Cybersecurity Challenges and Opportunities in a Smarter World

By
Pete Popov,
CEO, Konsulko Group

The combination of artificial intelligence (AI) and embedded systems is transforming the way the world interacts with technology. From smart cars and industrial automation to medical devices and intelligent home appliances, embedded systems have evolved from performing basic tasks into powerful, AI-enabled machines capable of learning, adapting, and making decisions in real-time.

However, with the introduction of AI comes an increase in potential cybersecurity vulnerabilities that must be addressed. Embedded systems combine hardware and software components, making them inherently vulnerable to attacks on both fronts.

Tech companies like NVIDIA have pioneered AI capabilities in embedded systems, and their hardware accelerates how AI can be used in everyday devices. However, this also brings heightened concerns over cybersecurity as cybercriminals continually seek ways to exploit vulnerabilities.

Yet, businesses can leverage AI to safeguard these systems and build more resilient, secure products. Here I discuss how developers can protect AI-enabled embedded systems and address the security challenges posed by the combination of AI and internet-connected devices.

AI-DRIVEN VULNERABILITIES IN EMBEDDED SYSTEMS

The integration of AI into embedded systems can introduce new security vulnerabilities. Embedded systems are often vulnerable to cyberattacks due to their constrained resources, physical accessibility, and long deployment lifecycles all of which can expose multiple points of attack.

The embedded security market has also grown significantly, with one report estimating that it will be worth nearly $10 billion by 2028. As these systems become more prevalent, there will be an increase in cybercriminals planning to take advantage of any vulnerabilities in these systems.

In one recent report, NVIDIA announced that Riva, its conversational AI platform, required a patch after two vulnerabilities were discovered. These issues affected certain versions of Nvidia Riva on Linux operating systems. Researchers who discovered these vulnerabilities warned that they could have resulted in significant financial losses, highlighting the importance of proactively testing systems.

These embedded systems offer incredible processing power, but this power can also serve as an entry point for attackers if not properly secured.

EVERYDAY DEVICES ARE AT RISK

As homes and businesses become smarter, the likelihood of cybercriminals attempting to access secure data by any means possible, including through smart devices, increases. A recent report found that around 57% of devices are vulnerable to hacking, thanks to a lack of encryption.

From voice-activated assistants like Amazon Alexa and Google Home, to smart

FIGURE 1
The medical industry is one sector that is heavily investing in AI.

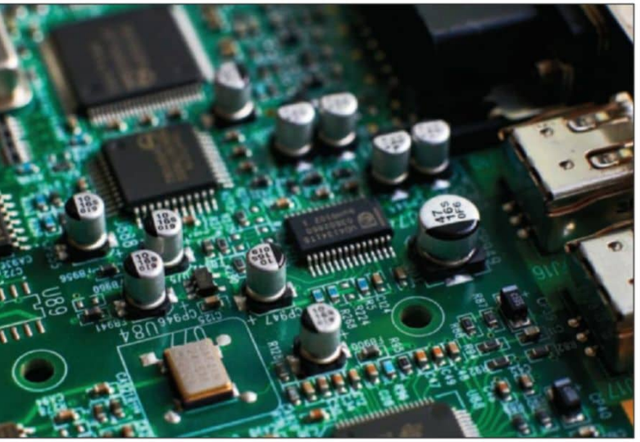


FIGURE 2
As AI evolves and becomes integrated into devices, security needs to start at the most basic level.

refrigerators and health monitors, these AI-driven devices have become a part of everyday life. But the same AI that allows these devices to function seamlessly can also be a weak link in the security chain.

For example, AI-powered security cameras or home automation systems can be hacked, allowing cybercriminals to access private footage or even control home security features.

In one instance, researchers discovered that AI-enabled cameras used for home security are vulnerable to hacking. These cameras are designed to distinguish between dogs walking in the neighborhood and criminals attempting to steal packages from porches. However, the camera could be hacked to confuse the AI and not trigger any alerts. While this was just one instance, this highlights potential vulnerabilities in these smart devices.

AI-powered infrastructure is also becoming increasingly common. NVIDIA recently announced that its technologies will be used to increase the capabilities of smart cities. This includes AI-powered traffic management systems. However, as these systems rely on a series of connected devices, cyberattacks could have widespread implications, affecting public safety.

Another sector that's increasingly investing in AI-enabled devices is the medical industry, with hospitals using smart devices to monitor heart rates, deliver insulin, or perform diagnostics. NVIDIA announced significant investments in the healthcare technology industry, with plans to bring AI into hospitals with physical robots to undertake a range of tasks, from X-ray imaging to linen delivery.

Health monitoring devices, including oximeters and peak flow meters, have also become incredibly popular, with the market for these devices expected to reach $270 billion by 2029. However, research has shown that they face risks such as spoofing, data injection and manipulation of settings.

If hackers gain control over these systems, they could change settings or even cause a device to malfunction, endangering patient safety.

ENSURE COMPREHENSIVE SECURITY FOR EMBEDDED SYSTEMS

As AI-enabled embedded systems evolve and become integrated into more devices for a wide range of purposes, the strategies for properly securing them also need to evolve. Traditional cybersecurity measures, such as firewalls, antivirus software, and data encryption, are not comprehensive enough to protect advanced AI systems.

At the foundation of embedded security is a chain of trust that begins at power-on and extends through the bootloader, operating system, application software, and over-the-air updates. Secure boot is critical to enforcing this trust. It ensures that only authenticated, untampered software is loaded at each stage of the boot process. If any link in the chain is compromised, the device will refuse to boot, preventing malicious control.

Using end-to-end encryption to safeguard data in transit ensures that communication between devices remains private and secure. Within the system, tools like dm-crypt and fscrypt protect data at rest by encrypting the underlying storage or filesystems. The list of existing open source technologies that can be used to secure a device is long.

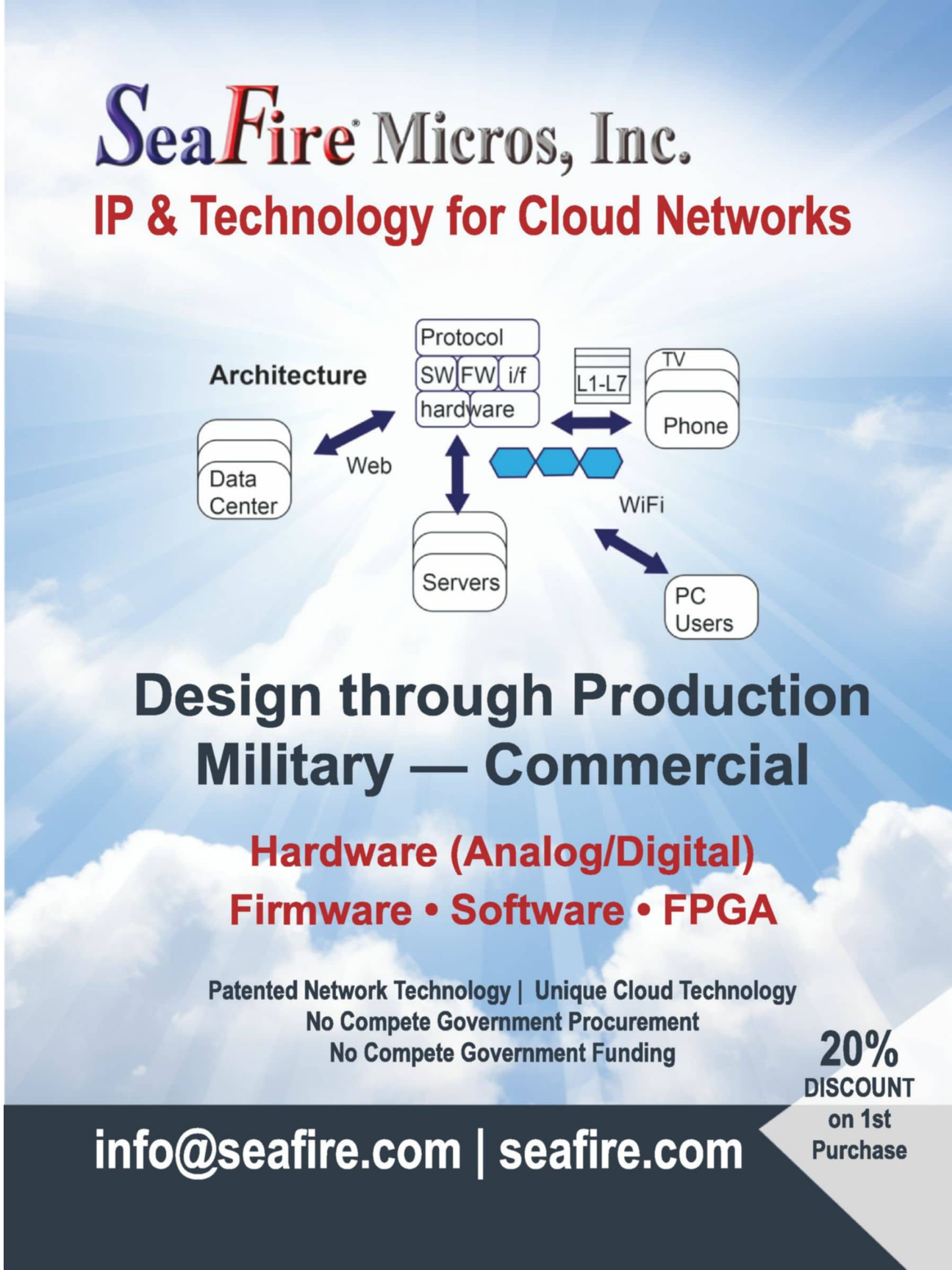
In addition to these traditional methods, today AI can also be part of the defense strategy. Embedded systems can use machine learning models to monitor real-time behavior and detect anomalous activity that may signal a threat. Unsupervised learning techniques such as clustering, autoencoders, and anomaly detection algorithms allow these systems to identify previously unknown attack patterns, providing a proactive layer of protection beyond static rule sets.

There is a vast array of applications that are made possible by using AI-enabled embedded systems. However, these systems require a multilayered defense approach, combining both traditional practices and modern security technologies. With the right architecture, developers can harness the potential of AI in embedded devices while maintaining user safety and trust.

RESOURCES
Konsulko Group
www.konsulko.com

ABOUT THE AUTHOR

Peter (Pete) Popov has over 25 years of experience in the high-tech industry in Silicon Valley in California. Pete holds a Bachelor of Science degree in electrical engineering from California Polytechnic State University, San Luis Obispo. A Linux user since 1995, Pete made Linux a full-time career in 2000 when he joined MontaVista Software as a kernel developer. Following this, he co-founded Embedded Alley Solutions, which was acquired by Mentor Graphics in 2009. In 2012 he went on to establish Konsulko Group, where he serves as CEO.



TS-9370

Low Power, High Performance SBC

The TS-9370 is a versatile and power-efficient industrial SBC built for rugged applications. Powered by the NXP i.MX9352 64-bit ARM® architecture plus two Cortex®-A53 processors operating up to 1.7 GHz, it delivers robust performance with low power consumption.

32 GB eMMC (upgradable)

2 GByte LPDDR4 RAM

MikroBUS socket
For MikroE Click Boards

Made in the USA
with global parts

embeddedTS

TS-4100

Ultra-Efficient and Fully Featured

The TS-4100 features an NXP i.MX 6UltraLite CPU, industrial temp range, FPGA, eMMC, and customizable I/O. Comes preloaded with Debian Stretch.

TS-7100-Z

As Small as Two Decks of Cards

Powered by an Arm® Cortex®-A7 i.MX6ul CPU, it runs efficiently at <2W, featuring Dual Ethernet, Bluetooth, WiFi, and NimbeLink/Digi sockets.

www.embeddedTS.com